MÉMOIRE PRÉSENTÉ

PAR

ARSÈNE SABAS

à l'Université du Québec à Trois-Rivières

Comme exigence partielle de la
Maîtrise en mathématiques et informatique appliquées

SYSTÈMES MULTI-AGENTS : UNE ANALYSE COMPARATIVE DES MÉTHODOLOGIES DE DÉVELOPPEMENT

Vers la convergence des méthodologies de développement et la standardisation des plateformes SMA

Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières

Décembre 2001

Toutes les choses que nous faisons extérieurement
n'ont aucun sens s'il n'y a pas une adéquation entre
ces choses et ce que nous pensons.

Capitaine

Résumé

Les systèmes multi-agents (SMA) sont actuellement très largement utilisés, particulièrement pour les applications complexes nécessitant l'interaction entre plusieurs entités. Dès lors, il est devenu impératif de s'investir dans les méthodologies orientées-agent et autres techniques de modélisation adéquates. Notre travail de recherche se place dans le contexte du génie logiciel et de l'intelligence artificielle distribuée où nous étudions plus spécifiquement, l'aspect génie logiciel dans le développement des SMA. Plusieurs méthodologies ont été proposées pour le développement des SMA. Ces méthodologies constituant soit une extension des méthodologies orientées-objet, soit une extension des méthodologies à base de connaissances, demeurent incomplètes. Par ailleurs, peu d'efforts ont été faits en matière de standardisation des plateformes SMA. Il apparaît donc évident que le développement des SMA reste encore un domaine ouvert. Le succès du paradigme orienté-agent exige des méthodologies systématiques pour la spécification et la conception des applications SMA. Notre travail de recherche constitue une approche de solution. Dans ce mémoire, nous proposons le cadre CaMuCCoSMA (Cadre Multidimensionnel de Critères pour la Comparaison des Méthodologies SMA). Ce cadre nous a permis de faire une étude comparative des principales méthodologies SMA existantes. Sur la base des résultats de cette étude comparative, nous proposons quelques suggestions pour la conception d'une méthodologie relativement complète de développement de SMA. Ces suggestions constituent une piste d'unification des méthodologies étudiées (dans le même esprit que UML). Enfin, pour rendre facile l'exploitation du cadre CaMuCCoSMA et la communication entre les différents acteurs de développement de SMA, nous proposons une description de CaMuCCoSMA, écrite dans la syntaxe CML (Conceptual Modeling Language).

Ce faisant, nous espérons rendre systématique la conception des applications SMA et contribuer à la standardisation des concepts et des plateformes SMA. Notre objectif à long terme est de concevoir une méthodologie relativement complète de développement de SMA. Ce travail de recherche a fait l'objet d'une publication [SAB01a] et d'une communication au 69e congrès de l'ACFAS [SAB01b].

Abstract

Because of the great interest in using multiagent systems (MAS) in a wide variety of applications in recent years, agent-oriented methodologies and related modeling techniques have become a priority for the development of large scale agent-based systems. The work we present here belongs to the disciplines of Software Engineering and Distributed Artificial Intelligence. More specifically, we are interested in software engineering aspects involved in the development of multiagent systems (MAS). Several methodologies have been proposed for the development of MAS. For the most part, these methodologies remain incomplete : they are either an extension of object-oriented methodologies or an extension of knowledge-based methodologies. In addition, too little effort has gone into the standardization of MAS methodologies, platforms and environments. It seems obvious, therefore, that software engineering aspects of the development of MAS still remains an open field. The success of the agent paradigm requires systematic methodologies for the specification, analysis and design of "non toy" MAS applications. The work we discuss here constitutes a step towards this goal. We present a comparative analysis of existing MAS methodologies which has also lead us to propose a unification scheme, much in the same spirit as that of UML, for MAS methodologies. To support our comparison task, we have designed a new grid called the CaMuCCoSMA (Multidimensional framework of criteria for the comparison of MAS methodologies). This specification is an initial unification scheme for these various methodologies. Lastly, to facilitate the exploitation of the CaMuCCoSMA framework and the communication between the various actors of the MAS development, we propose a description of CaMuCCoSMA, written with the CML syntax (Conceptual Modeling Language).

We hope to contribute to the convergence of MAS methodologies, as well as to the standardization of MAS platforms. Our long-term goal is the development of a rigorous and complete methodology for the analysis and design of MAS. This research work led to a publication [SAB01a] and a communication to the 69e congress of ACFAS [SAB01b].

Dédicaces

À

Mon père Sabas Alohou,

Ma mère Sabine Alohou,

Mon frère Étienne,

Mes autres frère et sœurs,

Je dédie ce mémoire.

Remerciements

Mes remerciements vont en direction de :

- Mon directeur de mémoire, Mourad Badri, professeur d'informatique au département de mathématiques et d'informatique à l'Université du Québec à Trois-Rivières. Je le remercie pour sa patience, ses précieux conseils, sa bonne volonté et sa disponibilité permanente.

- Mon co-directeur, Sylvain Delisle, professeur d'informatique et directeur du programme de maîtrise en mathématiques et informatique appliquées au département de mathématiques et d'informatique à l'Université du Québec à Trois-Rivières. Je le remercie pour son enthousiasme, sa bonne volonté, sa patience et sa disponibilité permanente.

  Je les remercie beaucoup pour leur soutien indéfectible.

- Robert Labarre, professeur d'informatique et ex-directeur du programme de maîtrise en mathématiques et informatique appliquées à l'Université du Québec à Trois-Rivières. Je le remercie pour sa rigueur et sa sympathie.

- Mes professeurs du département de mathématiques et d'informatique pour leurs conseils et leur sympathie.

- Lise Branchaud, la secrétaire du programme de maîtrise, pour sa disponibilité permanente et l'aide qu'elle m'a apportée dans la mise en forme générale de ce document.

- Le groupe DAMAS (Datamining Agent MultiAgent Systems) et son directeur, Brahim Chaïb-Draa, professeur titulaire d'informatique au département d'informatique de l'Université Laval, pour la lecture de ce mémoire et les corrections apportées.

- Bernard Moulin, professeur titulaire d'informatique au département d'informatique de l'Université Laval, pour avoir jeté un coup d'œil sur les conclusions de ce travail.

- Mon père Sabas Alohou et mon jeune frère Étienne défunts, ma mère Sabine Alohou, mes frère et sœurs Agathe, Martine, Faustine, Flore et Victorien pour toute l'affection et services qu'ils m'ont toujours apportés. C'est le lieu de leur dire merci.

- Hervé Koussé pour son amitié désintéressée.

- Richard Okambawa pour avoir accepté, malgré ses occupations, de lire ce mémoire et d'y apporter des corrections.

- Tous mes amis qui, de près ou de loin, de part leur soutien moral ont contribué à l'aboutissement de ce projet.

Table des matières

Liste des figures

Liste des tableaux

Première partie

Introduction

Chapitre 1
Introduction

Ce chapitre d'introduction a pour but de définir les mots-clés et quelques concepts qui dominent notre travail de mémoire. La définition du sens précis dans lequel nous utilisons constamment ces termes nous permettra sans doute d'éviter des malentendus.

1.1 Génie logiciel

Nous ne pouvons définir l'expression Génie Logiciel (GL) sans préciser le sens du mot logiciel lui-même.

1.1.1 Logiciel

Le logiciel est un système informatique contenant les différentes formes de programmes (source, éditable, exécutable, etc.), les fichiers de données qu'ils utilisent, la documentation sur la conception, sur la mise en œuvre, sur les modifications associées et le manuel d'utilisation. La finalité du produit logiciel est de résoudre des problèmes, de faire des simulations, de l'intégrer à d'autres produits logiciels, etc.

1.1.2 Génie logiciel

Le Génie Logiciel (GL) est né, il y a environ trois décennies sous le nom de "Software Engineering" pour résoudre la problématique qui était la suivante :

- le logiciel produit n'était pas fiable : il arrive très souvent que l'on s'aperçoive que le logiciel développé ne corresponde pas à la demande;
- la réalisation du logiciel dans le délai et le budget prévus était difficile.

Plusieurs définitions ont été attribuées à cette expression. D'une manière générale, le GL est définit comme étant l'ensemble des connaissances, des procédés et des acquis scientifiques et techniques mis en application pour la conception, le développement, la vérification, la documentation de logiciels et la maintenance, dans le but d'en optimaliser la production, le

support et la qualité [GRA01]. C'est l'art de développer le produit logiciel plus correct et plus flexible, tout en restant raisonnablement dans les délais et budget prévus.

L'objectif du GL est de réduire le temps et le coût, de discipliner les phases de développement des logiciels et d'assurer la qualité de ceux-ci.

«La construction de logiciels de qualité n'est possible que par la définition scientifique et la mise en action d'une épistémologie pragmatique, fondée sur des méthodes, langages, modèles et outils tenant compte de l'état de l'art et de celui des pratiques, et aussi détachée que possible du seul court terme, malgré l'évolution très rapide de l'informatique» [THE88].

La pratique du Génie Logiciel est particulièrement indispensable et incontournable pour des industries développant des logiciels, surtout si ces derniers sont d'une grande taille ou si leur utilisation présente des risques pour la vie humaine ou la propriété.

1.2 Intelligence artificielle distribuée

L'expression Intelligence Artificielle (IA) est employée pour la première fois (1955-1970) par John McCarthy. Il fonde l'Intelligence Artificielle sur le postulat mécaniste qui veut que toute activité intelligente soit modélisable et reproductible par une machine.

« L'IA a pour but de faire exécuter par l'ordinateur des tâches pour lesquelles l'homme, dans un contexte donné, est aujourd'hui meilleur que la machine» [ALL94].

L'Intelligence Artificielle Distribuée (IAD) est un sous-domaine de Intelligence Artificielle qui s'occupe des situations où plusieurs systèmes interagissent pour résoudre un problème commun [MOU96b]. L'IAD se divise en deux branches principales :

- la Résolution Distribuée de Problèmes (RDP) qui étudie comment distribuer des compétences au niveau de chaque partie du système, de façon à ce qu'il soit globalement plus compétent que chacune de ses parties;
- la Simulation des Systèmes Complexes (SSC), qui concerne plus particulièrement les Systèmes Multi-Agents (SMA). Les SMA traitent le comportement d'un ensemble d'agents autonomes qui essaient de résoudre un problème commun.

La différence notable entre la RDP et les SMA est que la RDP possède une approche descendante («top-down») et les SMA une approche ascendante («bottom-up»).

1.3 Systèmes multi-Agents

Bien qu'il y ait peu de consensus autour des concepts agents et SMA, plusieurs chercheurs ont des définitions qui convergent vers celles-ci :

1.3.1 Agents

D'après [WOO00a]. Un agent est un système informatique encapsulé situé dans un environnement dans lequel il est capable d'effectuer une action flexible et autonome, compatible aux objectifs de la conception. Les agents sont :

- des entités clairement identifiables de résolution de problèmes avec des bornes et des interfaces bien définies;
- situés dans un environnement particulier ; ils reçoivent des entrées liées aux états de cet environnement par des capteurs et agissent sur cet environnement par des émetteurs;
- destinés à atteindre un objectif spécifique;
- autonomes et responsables de leur comportement;
- capables d'adopter un comportement flexible pour résoudre des problèmes selon les objectifs de la conception; ils sont réactifs (capables de s'adapter aux changements d'état de leur environnement) et proactifs (capables d'adopter un nouvel objectif);
- capables dans un univers multi-agents, de communiquer, coopérer, se coordonner, négocier les uns avec les autres.

La figure 1.1 donne, de façon générale, l'architecture interne d'un agent.

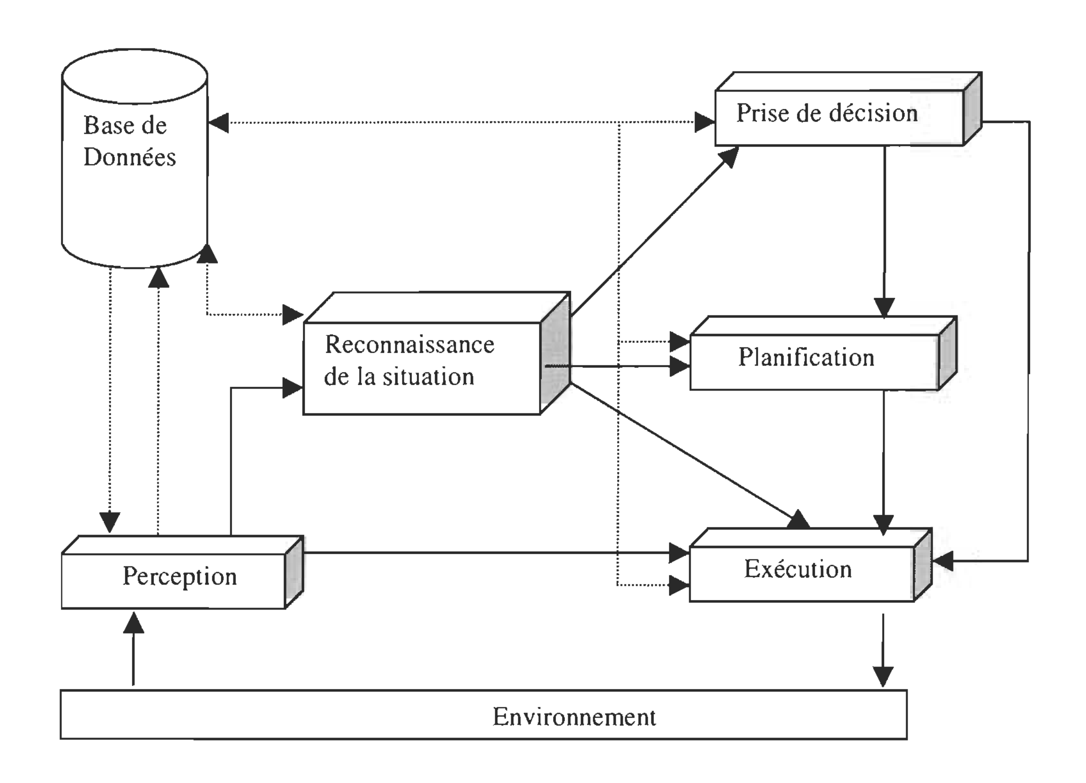


Figure 1.1 : Architecture d'agent [COT99]

Lorsqu'un agent perçoit une situation dans l'environnement, il essaie de la reconnaître. Si la situation lui est familière, il peut enclencher un processus de planification afin de résoudre le problème. Il peut aussi reconnaître la situation en terme d'action et donc, passe à l'exécution de la tâche (Reconnaissance- Exécution). Lorsque l'agent perçoit des situations qu'il connaît très bien, il peut faire intervenir son comportement réactif en passant directement à l'action (Perception-Exécution). S'il ne peut pas résoudre un problème (situation non-familière), il engage un processus de coopération pour demander de l'aide aux autres agents (Reconnaissance-Prise de décisions).

1.3.2 Systèmes multi-Agents

Les systèmes multi-agents mettent en œuvre des agents homogènes et hétérogènes ayant des buts communs ou distincts. Ils sont dynamiques.

Un système multi-agent est un système distribué composé d'un ensemble d'agents qui interagissent le plus souvent, selon des modes de coopération, de concurrence ou de coexistence.

Selon [CHA99] un SMA est généralement caractérisé par :

1. chaque agent a des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent a un point de vue partiel;
2. il n'y aucun contrôle global du système multi-agents;
3. les données sont décentralisées;
4. le calcul est asynchrone.

Le principe récursif définit un SMA à un niveau supérieur d'abstraction comme étant un agent. Une vue canonique d'un SMA est donnée dans la figure 1.2.

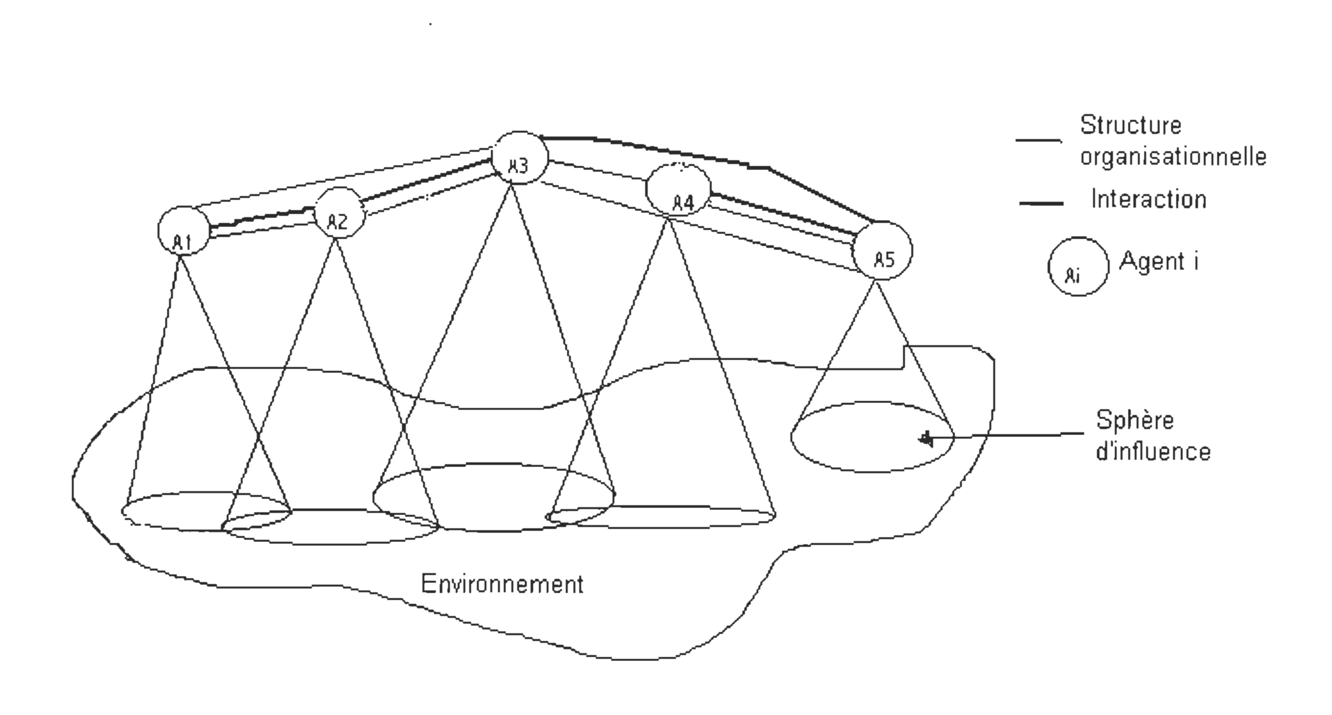


Figure 1.2 : Vue canonique d'un SMA [WOO00a]

1.3.3 Environnement

Selon [PES97], l'environnement peut être considéré comme la représentation du monde dans lequel les agents se situent. L'environnement est modifiable par les agents, soit de façon

globale, soit en faisant la distinction entre objets passifs (soumis aux actions des agents) et entités actives (les agents) [FER95].

1.3.4 Interaction

Ferber [FER95] définit les interactions comme étant la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. L'interaction entre agents s'effectue par la communication, les actes de langages et les protocoles d'interaction. Les agents interagissent entre eux. Pour atteindre son objectif ou pour améliorer la coordination des actions, un agent peut demander des services à un autre agent. Il y a deux éléments qualitatifs qui caractérisent les interactions entre agents :

- les interactions se produisent à un niveau élevé par un langage de communication, en fonction du temps, de l'objectif à atteindre et de la nature des agents;
- comme les agents sont des solutionneurs de problèmes flexibles dans un environnement au delà duquel ils ont uniquement un contrôle partiel, les interactions devraient également être flexibles. Ainsi, les agents ont besoin d'un appareil de calcul pour prendre des décisions, dans un contexte dépendant de certains facteurs, au sujet de la nature et de la portée de leurs interactions et pour provoquer des interactions qui n'ont pas été prévues lors de la conception.

Les interactions définissent un contexte organisationnel qui définit la nature des rapports entre agents et qui influence le comportement individuel des agents. Il est donc important de représenter explicitement ces rapports qui sont sujets à un changement continu. Les rapports existants évoluent dans le temps et de nouvelles relations se créent.

Pour faire face à cette variété et cette forme dynamique de rapports, les chercheurs ont :

- conçu des protocoles rendant capable la formation du contexte organisationnel;
- spécifié des mécanismes permettant aux agents d'agir ensemble de façon cohérente;
- et développé des structures pour caractériser le macro-comportement de l'ensemble.

1.3.5 Organisation

Étant donné que les SMA peuvent être considérés comme une société d'agents coopérant ensemble pour accomplir collectivement un objectif donné, il est nécessaire de résoudre un problème d'organisation, généralement de façon dynamique.

Selon Fox [FOX81], on peut définir une organisation comme une structure décrivant comment les membres de l'organisation sont en relation et interagissent afin d'atteindre un but commun.

L'autonomie et le comportement proactif des agents constituant les SMA suggèrent que la conception de ces applications peut être réalisée en imitant le comportement et la structure des organisations humaines, car l'une des missions principales des SMA est de supporter et/ou de contrôler des organisations du monde réel. Un SMA peut, par exemple, aider à contrôler les activités du commerce électronique. Selon [ZAM00], la perspective organisationnelle conduit à une caractérisation générale d'un SMA décrite dans la figure 1.3.

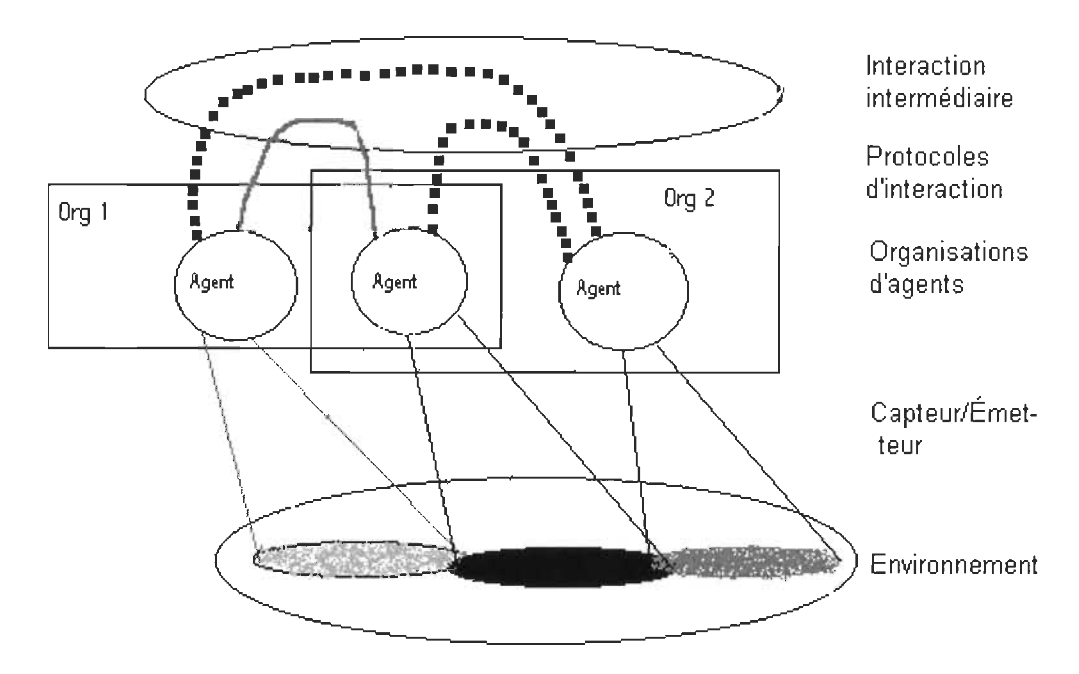


Figure 1.3 : Caractérisation d'un SMA [ZAM00]

Le système peut être décomposé en des sous-organisations distinctes. Un agent peut jouer un ou plusieurs rôles tout en coopérant et en respectant ses sous-organisations. Les interactions

entre agents apparaissent via les capteurs et les émetteurs. Le rôle d'un agent détermine la portion de l'environnement dans laquelle il peut recevoir et émettre des données.

Nous définirons dans le chapitre 6 les principaux types de structures organisationnelles dans les SMA que nous avons pu recenser dans la littérature.

1.3.6 La nature de la complexité des systèmes logiciels agents

La nature de certains systèmes informatiques (industriels) est complexe et est caractérisée par la taille de leurs interactions internes. Cette complexité est directement liée au type d'application auquel le système est destiné. Mais elle possède un certain nombre de points réguliers dans tous les types d'applications :

- elle a souvent une structure hiérarchique dynamique;
- le choix des composantes primitives du système est relativement arbitraire et est défini beaucoup plus selon les directives et les objectifs de l'utilisateur;
- on ne peut prévoir toutes les interactions lors de la conception.

Le but du GL est de fournir des structures et techniques puissantes pouvant permettre de maîtriser facilement cette complexité. BOOCH [WOO00a] a identifié trois de ces techniques :

1. La décomposition

Il s'agit de décomposer le système en des sous-systèmes de façon à ce que :

- à un niveau quelconque donné, les sous-systèmes coopèrent pour résoudre la fonctionnalité de leurs parents. À chaque sous-système correspondra un ou plusieurs objectifs;
- Les sous-systèmes soient autonomes, responsables de leurs actions, réactifs et proactifs;

Cette décomposition diminue le couplage et les synchronisations sont complètement maîtrisées à travers les interactions entre sous-systèmes;

Il est naturel de modulariser un système complexe en termes d'interactions et de composantes autonomes ayant des objectifs spécifiques à atteindre.

2. L'abstraction

L'abstraction est le processus permettant d'avoir un modèle simplifié. Le problème à caractériser est composé de : sous-systèmes, composantes de sous-systèmes, interaction, organisation de rapports.

Les sous-systèmes correspondent aux agents. Les propriétés des composantes des sous-systèmes sont des caractéristiques des agents. Les systèmes complexes exigent que des composantes soient regroupées dans une unité conceptuelle.

3. L'organisation

L'organisation est le processus permettant d'identifier et de contrôler les rapports entre les diverses composantes. Les représentations explicites sont faites d'organisations de rapports et de structures.

1.3.7 Approche orientée-agent et approche orientée-objet

Au regard des caractéristiques d'agent, il apparaît que l'approche orientée-agent dans le développement de logiciel consiste en une décomposition du problème en agents ayant des interactions, une autonomie, et un objectif spécifique à atteindre. Les concepts clés d'abstraction liés à un système orienté-agent sont : agent, interaction, organisation.

Bien qu'il existe une similarité superficielle entre objet et agent, la terminologie objet n'est pas adaptée aux systèmes agents :

- les objets sont généralement passifs alors que les agents sont permanents actifs;
- les agents sont autonomes et responsables de leurs actions alors que les objets n'en sont toujours pas;
- on ne peut prévoir tous les comportements des agents dans les systèmes;
- l'approche orientée-objet ne fournit pas un ensemble adéquat de concepts et de mécanismes pour modéliser les systèmes complexes dans lesquels les rapports évoluent dynamiquement;
- certains chercheurs définissent un agent comme un objet actif ayant une autonomie et un objectif.

1.3.8 Agent holonique

Les systèmes holoniques (figure 1.4) ont été proposés par A. Koestler (en 1969) pour décrire les systèmes biologiques et sociaux. Cette proposition fait suite à une étude sur le fonctionnement des différentes sociétés non seulement humaines (communistes, capitalistes, dictatoriales, démocratiques) mais aussi animales et végétales (l'arbre ou la plante sont alors vus comme une société composée d'entités). C'est une approche systémique des organisations. Chaque partie du système, appelée holon, est elle-même décomposable en holons. Le principe holonique veut que chaque partie soit : stable pour pouvoir faire face à des perturbations; autonome pour pouvoir agir d'elle-même afin de réaliser son objectif; coopérante pour se lier à d'autres parties afin de réaliser l'objectif commun du système. Ces systèmes font l'objet d'un courant de recherche actuellement dans le domaine de la robotique et des systèmes manufacturiers. On peut, en effet, concevoir une entreprise comme une entité composée de sous-entités qui, idéalement, devraient être stables et avoir des degrés d'autonomie et de coopération équilibrés [ADA99a].

L'avantage d'un système holonique est qu'il possède une architecture générique et récursive. Du point de vue agent, un agent holonique a :

- une identité (nom, type, état);
- des connaissances (traitement, autres acteurs);
- un comportement (conçoit, reçoit, envoie, agit, examine);
- respecte un ensemble de règles. Koestler [KOE69] propose un ensemble de 65 règles décrivant les notions de dualité coopération-autonomie, de communication et d'architecture. Ces règles sont regroupées autour de dix ensembles définissant les systèmes holoniques [ADA99b].

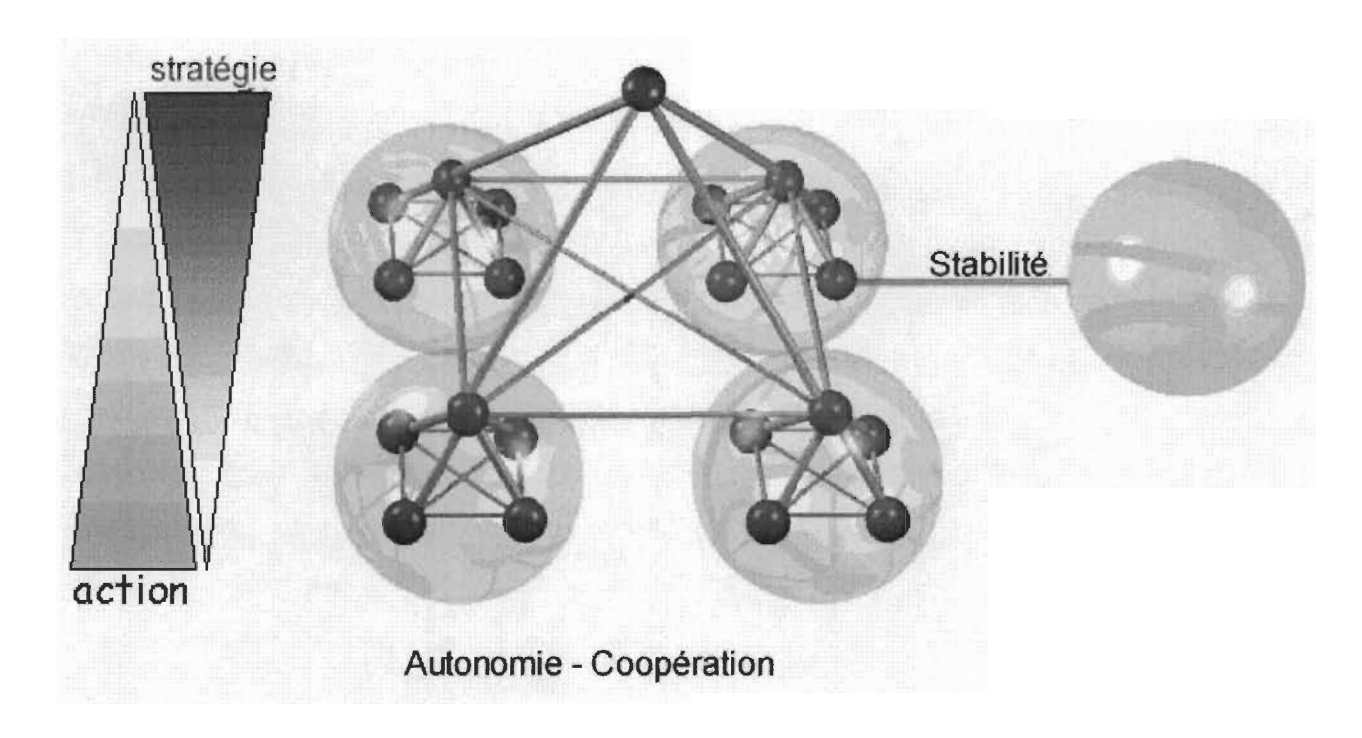


Figure 1.4 : Un exemple de système holonique [ADA99c]

1.3.9 Facilitateurs – médiateurs – courtiers – tableau noir

Des mécanismes pour la publicité, la découverte, l'utilisation, la présentation, la gestion et la mise à jour des services et des informations fournis par les agents sont nécessaires. Pour réaliser ces mécanismes, des agents intermédiaires sont proposés [ROO99]. Les agents intermédiaires sont des entités auxquelles d'autres agents publient leurs capacités, et qui ne sont ni demandeurs ni fournisseurs. L'avantage de tels agents, est qu'ils permettent à un SMA d'opérer de façon robuste face à l'apparition, à la disparition et à la mobilité des agents.

Les différents types d'agents intermédiaires [ROO99] sont :

- **Les facilitateurs**

Ce sont des agents auxquels d'autres agents abandonnent leur autonomie en échange des services de ces agents (facilitateurs) [ROB99]. Les facilitateurs ont été utilisés dans le modèle dynamique d'interaction proposé par Ribero [RIB00]. Ils possèdent un répertoire des services et capacités des agents et sont capables d'aider dans l'aiguillage du flux d'information et dans la mise en relation des agents. Leur utilisation simplifie les connaissances dont les agents ont besoin à propos des autres agents. Ils peuvent se cantonner à faire appel aux facilitateurs qui, à

leur tour, iront fournir les informations sur les autres agents ou sur les services dont ils ont besoin [RIB00]. Ce concept a été également utilisé dans le langage KQML [FIN93] [LAB97].

- **Les médiateurs**

Il existe plusieurs approches à l'application du concept de médiateur dans les SMA. Un médiateur est un agent qui exploite les connaissances codées sur un ensemble (ou sous-ensemble) de données pour créer des informations pour un niveau d'application supérieur [WIE98]. Selon [GEO95], un agent médiateur est un agent logiciel qui gère avec un dossier les interactions des agents personnels présents dans un même lieu.

- **Les courtiers**

Ce sont des agents qui reçoivent des demandes et exécutent des actions en utilisant des services d'autres agents, en conjonction avec leurs propres ressources [DEC96].

- **Les apparieurs et les pages jaunes**

Ce sont des agents qui fournissent des services de pages jaunes pour les autres agents. Les agents peuvent publier leurs compétences (services) et consulter celles des autres agents [RIB00]. Les apparieurs assistent les demandeurs pour trouver des agents fournisseurs de service, en se basant sur les compétences publiées [BRA97], [DEC96].

- **Tableau noir**

Ce sont des agents repositoires qui reçoivent et maintiennent des demandes pour le traitement d'autres agents [NÜ86], [COH94].

1.3.10 Langages de communication entre agents

Il y a deux principales approches pour concevoir un langage de communication entre agents [GEN97]. La première approche est une approche procédurale où la communication est basée sur le contenu exécutable. Ceci pourrait être réalisé en utilisant les langages de programmation tel que Java par exemple [ARN98]. La seconde approche est une approche déclarative, où la communication est basée sur des déclarations telles que des définitions et des hypothèses.

Selon [ROB99], les approches déclaratives ont été préférées pour la conception de langages de communication entre agents, probablement à cause des limitations sur les approches procédurales (le contrôle du contenu exécutable est difficile).

La plupart des implémentations des langages déclaratifs sont basées sur des actes illocutoires, tels qu'une demande ou une commande; les actions sont communément appelées performatives.

Vanderveken a travaillé avec Searle dans la classification des actes de langage et ultérieurement dans l'axiomatisation de la théorie des actes de langage [VAN90]. Il pose qu'un acte de langage A est de la forme : A = F(P) où F est une force illocutoire appliquée à un contenu propositionnel P.

La force illocutoire est présente dans une phrase sous la forme d'un verbe à la première personne de l'indicatif [RIB00]. Par exemple, pour demander à une personne si elle possède un crayon (as-tu un crayon?), l'acte de langage doit être : je demande si tu as un crayon; le performatif ici est le verbe demander.

Un des langages déclaratifs d'agents les plus utilisés est KQML.

KQML

Le langage KQML (Knowledge Query and Manipulation Language) [FIN93] [LAB97] a été développé au sein du groupe de travail «External Interfaces Group» du projet «Knowledge Sharing Effort» financé par l'ARPA («Advanced Research Projects Agency») [PAT92]. Ce langage peut être considéré comme composé de trois couches : une couche de communication qui décrit le niveau des paramètres de communication; une couche de message qui contient un performatif et indique le protocole d'interprétation (les performatifs indiquent quel type d'acte de langage l'agent désire employer et définissent les opérations possibles que les agents peuvent demander sur les connaissances et les objectifs des autres agents [PAT92]); et une couche du contenu qui contient des informations relatives au performatif soumis.

```
( performatif
                    : paramètre «mot»
                    : paramètre «mot»
                    ...
                    : content  «expression»
```

Figure 1.5 : Structure globale d'un message KQML [RIB00]

(Register
    : sender   agent A
    : receiver   agent B
    : reply-with   message 2
    : language   common_language
    : ontology   common_ontology
    : content   '(service Provision Manufacturing :
                    TaskDecomposition)'

Exemple d'un message KQML [RIB00]

Les performatifs peuvent être accompagnés de paramètres qui composent le contenu du message devant être transmis. Les paramètres sont opérationnels, leur utilisation dépend du performatif employé [RIB00].

1.4 Conclusion

Dans ce chapitre, nous avons défini les principaux concepts utilisés tout au long de ce mémoire. Ces définitions vont permettre au lecteur de comprendre le sens précis dans lequel nous les avons utilisés.

Chapitre 2

Hypothèses et objectifs de travail

Dans ce chapitre, nous présentons le contexte général de nos travaux de recherche et les motivations qui les sous-tendent. Nous présentons ensuite les objectifs et la démarche suivie pour les atteindre. Nous terminons ce chapitre par une description de l'organisation du reste de ce manuscrit.

2.1 Contexte de la recherche

Si les systèmes multi-agents (SMA) sont actuellement largement acceptés à une grande échelle d'applications diverses, des méthodologies orientées-agent et des techniques de modélisation adéquates sont devenues essentielles. Nos travaux de recherche se placent dans le contexte du Génie Logiciel et de l'Intelligence Artificielle Distribuée où nous étudions, plus spécifiquement, l'aspect génie logiciel dans le développement des SMA. Plusieurs méthodologies ont été proposées pour le développement des SMA. Ces méthodologies constituant soit une extension des méthodologies orientées-objet, comme par exemple Agent Modeling for System of BDI agents [KIN96], soit une extension des méthodologies à base de connaissance, comme par exemple CoMoMAS [NOR96]. Cependant, ces différentes méthodologies ne fournissent pas de techniques adéquates pour modéliser les caractéristiques spécifiques aux agents telles que leur état mental et leur comportement social dans un SMA. D'autres méthodologies, comme GAIA [WOO00b], ont vu le jour mais restent néanmoins toujours incomplètes. Elles ne prennent véritablement pas en compte, par exemple, la vérification des SMA. Nous avons aussi constaté qu'il y a eu peu d'efforts sur la standardisation d'architecture SMA. Les plateformes SMA, développées séparément par des groupes de recherche et d'industriels, diffèrent par leur architecture, les langages utilisés, etc. Il apparaît, donc, évident que le développement des SMA reste encore un domaine ouvert. Le succès du paradigme orienté-agent exige des méthodologies systématiques pour la spécification et la conception des applications SMA. Nos travaux de recherche constituent une approche de solution. Il s'agit, en effet, d'une étude comparative des méthodologies SMA existantes en vue

d'une spécification d'un cahier de charges pour la conception d'une méthodologie complète répondant aux caractéristiques des SMA et aux principes du Génie Logiciel.

2.2 Motivations

La réussite des applications fondées sur les SMA développés dans plusieurs domaines d'application comme la simulation, le contrôle de processus, le contrôle du trafic aérien, la gestion d'informations, le commerce électronique, la gestion des processus d'une entreprise, la surveillance des patients, les soins, les jeux, le théâtre et le cinéma interactifs, démontre que les SMA paraissent être l'approche la plus adaptée à un nombre important de domaines [JEN98]. L'extension des domaines d'application impose le développement de SMA chaque fois plus performants et, par conséquent, chaque fois plus importants et plus complexes à implémenter. La maîtrise de la complexité des SMA impose une certaine discipline dans les méthodologies de développement de ces derniers. Le besoin de méthodes et d'outils permettant la réalisation rapide et fiable des SMA se fait nettement sentir non seulement dans la communauté SMA elle-même, mais aussi pour permettre l'usage du paradigme agent pour un public plus large encore.

Le point de départ de nos travaux de recherche est d'examiner (entre autres) l'utilisation de méthodologies générales de développement exportées vers l'IAD.

2.3 Objectifs scientifiques

Les objectifs que nous avons poursuivis au cours de nos travaux de mémoire sont les suivants :

- faire une étude comparative des méthodologies de développement de SMA;
- déboucher sur une spécification complète d'un cahier de charges en vue de la conception d'une méthodologie répondant aux caractéristiques des SMA et aux principes du GL;
- participer à la convergence des concepts et méthodologies SMA;

2.4 Démarche suivie

De prime abord, nous avons réalisé une étude bibliographique. Cette étude nous a permis d'identifier les principales méthodologies (représentatives) de développement de SMA.

Pour faire une analyse comparative de ces méthodologies, nous avons senti le besoin d'un outil qui puisse la supporter. Nous avons donc conçu le cadre CaMuCCoSMA (Cadre Multidimensionnel de Critères pour la Comparaison des Systèmes Multi-Agents). Avec ce cadre, nous avons réalisé l'étude comparative des méthodologies SMA que nous avons pu recenser.

Sur la base des résultats issus de la comparaison, nous avons élaboré un cahier de charges qui constitue une piste d'unification des méthodologies étudiées, en vue de la conception (dans le futur) d'une méthodologie relativement complète de développement de SMA, qui prendrait en compte toutes les dimensions de notre cadre.

Enfin, pour faciliter l'exploitation du cadre CaMuCCoSMA, nous avons proposé dans l'annexe une description de ce cadre dans la syntaxe CML (Conceptual Modeling Language) [SCH94a].

2.5 Organisation du manuscrit

Le reste de ce manuscrit est composé de trois parties :

- la deuxième partie comporte trois chapitres. Elle couvre un état de l'art sur les principales méthodologies de SMA (chapitre 3), présente quelques outils supportant ces méthodologies (chapitre 4) et les plateformes SMA existantes (chapitre 5);

- la troisième partie, constituée d'un seul chapitre, décrit en détail le cadre CaMuCCoSMA que nous avons conçu et qui nous a permis d'analyser les méthodologies;

- la quatrième partie est composée de quatre chapitres. Dans le chapitre 7, nous avons fait une analyse comparative des principales méthodologies SMA selon le cadre CaMuCCoSMA. Le chapitre 8 présente une tentative d'unification des méthodologies SMA étudiées. Nous proposons un cahier de charges qui servirait d'une piste pour la conception d'une méthodologie relativement complète de SMA. Le dernier chapitre présente les conclusions de ce travail.

Pour faciliter l'exploitation du cadre CaMuCCoSMA et permettre une bonne communication entre les différents acteurs de développement de SMA, nous avons proposé dans l'annexe, une description de ce cadre dans une syntaxe simple, celle de CML (Conceptual Modeling Language) [SCH94a].

Deuxième partie

État de l'art

Chapitre 3

Les principales méthodologies SMA

Dans ce chapitre, nous allons décrire brièvement les principales méthodologies SMA que nous avons recensées dans la littérature. Il y a, parmi celles-ci, certaines sur lesquelles nous ne possédons pas assez d'informations nous permettant de faire une étude détaillée. Il nous semble cependant que les méthodologies que nous avons étudiées en détail, dans la quatrième partie de ce document, sont les principales et sont les plus représentatives des méthodologies SMA existantes. Les détails de ces principales méthodologies se trouvent au chapitre 7 de ce document.

Les méthodologies SMA existantes constituent soit une extension des méthodes orientées-objet, soit une extension des méthodes à base de connaissance. Certaines sont conçues pour un contexte particulier. Nous listerons, dans la suite du document, ces principales méthodologies tout en les décrivant succinctement.

3.1 Les méthodologies constituant une extension des méthodes orientées-objet

Les principaux avantages de ces approches sont :

- un agent peut être considéré comme un objet actif ayant un état mental, un objectif et une autonomie;
- les deux paradigmes (objet et agent) utilisent l'envoi de messages pour la communication, et peuvent utiliser l'héritage et l'agrégation pour définir leurs architectures;
- les méthodes orientées-objet sont populaires, en ce sens que plusieurs d'entre elles ont été utilisées avec succès dans l'industrie. L'expérience et le succès liés à cette utilisation peuvent faciliter l'intégration de la technologie agent;
- les modèles objet, dynamique et fonctionnel de ce paradigme peuvent être utilisés pour décrire les agents;

- les cas d'utilisation et les diagrammes de collaboration peuvent être utilisés dans l'identification des agents.

Mais, les méthodes orientées-objet telles quelles ne fournissent pas de techniques adéquates pour caractériser certains aspects spécifiques aux agents, tels que : leur état mental, le type des messages, le protocole de négociation, leur dimension sociale.

Dans ce qui suit, nous présentons des méthodes mieux adaptées aux agents.

3.1.1 La méthodologie GAIA

Selon [WOO00b], GAIA est le nom donné aux hypothèses formulées par l'écologiste James Lovelock selon lesquelles les organismes vivants du monde peuvent être compris comme des composantes d'une seule entité, lesquelles composantes régularisent l'environnement mondial. Les systèmes auxquels on peut appliquer cette méthodologie contiennent un petit nombre d'agents (moins de 100). Les auteurs de GAIA considèrent un SMA comme une société ou une organisation artificielle ou encore un ensemble institutionnalisé de rôles. Cette idée vient du fait que dans une compagnie, il y a un poste de président, un poste de vice-président, etc. L'instanciation (qui peut-être dynamique) de ces postes est fonction des individus qui sont présents à l'époque.

GAIA se compose de deux phases principales (figure 3.1) :

- la phase d'analyse qui produit deux modèles : le modèle de rôles et le modèle d'interactions;
- la phase de conception qui transforme les modèles abstraits de la phase d'analyse en modèles moins abstraits. Le processus de conception se base sur les trois modèles modèle d'agent, modèle de service et modèle de relation.

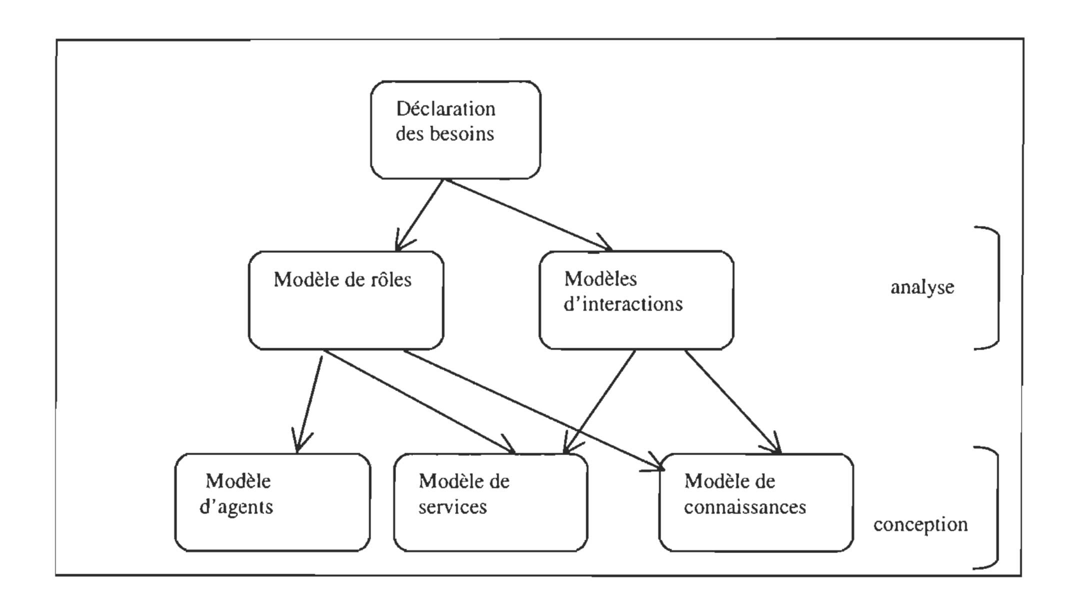


Figure 3.1 : Les modèles de GAIA [WOO00b]

3.1.2 Multiagent Systems Engineering (MaSE)

Les auteurs de cette méthodologie [SCO99] définissent un agent comme un objet actif ayant un objectif et un langage de communication. Cette méthodologie utilise les techniques de OMT (Object Modeling Technique) [RUM91] ou de UML (Unified Modeling Language) [MUL97] avec quelques caractéristiques de plus et quelques modifications de la sémantique du paradigme objet pour pouvoir capter les concepts d'agent et les comportements coopératifs des agents. Elle utilise deux langages pour décrire les agents et les SMA : Agent Modeling Language (AgML) et Agent Definition Language (AgDL).

Les différentes phases de MaSE (figure 3.2) sont :

- Conception du domaine (Domaine Level Design);
- Conception d'agents (Agent Level Design);
- Conception de composantes (Component Design);
- Conception du Système (System Design).

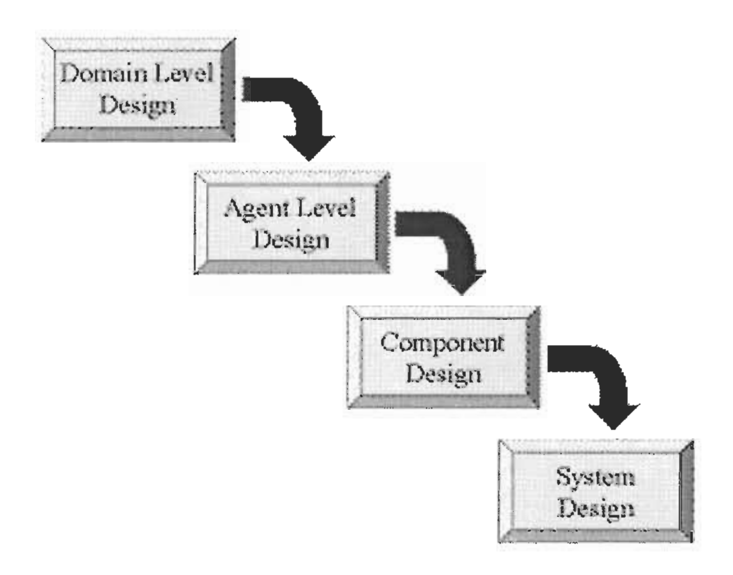


Figure 3.2 : Les différentes phases de MaSE [SCO99]

3.1.3 An Agent-Oriented Methodology : High-Level and Intermediate Models (HLIM)

La méthodologie HLIM [EAL99] fait ressortir, à travers deux phases, les aspects des agents tels que : leurs objectifs, leurs plans, leurs croyances et les rapports entre ces agents. Comme les techniques orientées-objet capturent les objets dans le système, leurs attributs, leur structure et leurs liens, HLIM capture les agents, leurs attributs, leur structure et leurs liens.

La phase de découverte, qui est une phase exploratoire du domaine, est la première phase de HLIM qui produit le modèle High-Level Model. Ce modèle capture la structure et le comportement du système. La deuxième phase de définition produit les 4 modèles suivants :

- Internal Agent Model;
- Relationship Model;
- Conversational Model;
- Contract Model.

La figure 3.3 donne une vue globale des différents modèles de HLIM avec leur séquencement.

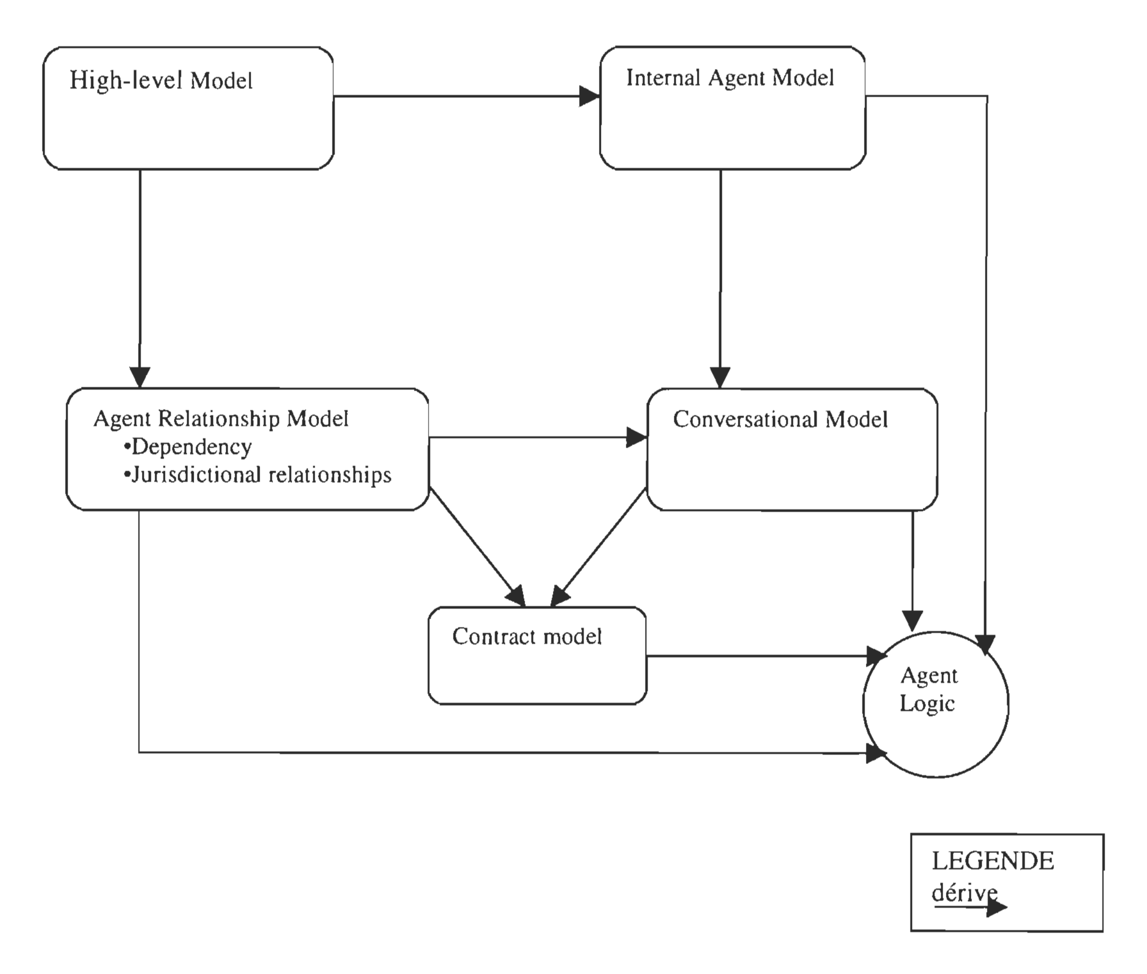


Figure 3.3 : Les modèles de HLIM [EAL99]

3.1.4 A Methodology and Modeling Technique for Systems of BDI agents (MMTS)

Cette méthode [KIN96] définit deux principaux niveaux pour la modélisation des agents BDI :

- le niveau externe (qui utilise la notation OMT) consiste en la décomposition du système en agents et la définition de leurs interactions. Ceci se traduit à travers deux modèles : le modèle d'agent et le modèle d'interactions ;

- le niveau interne fait ressortir la modélisation de chaque classe agent BDI à travers trois modèles : le modèle de croyance, le modèle d'objectif et le modèle de plan.

Pour plus de détails sur MMTS, référer au chapitre 7 de ce document, [KIN96] et [DAV97].

3.1.5 Agent-Oriented Analysis and Design (AOAD)

Tous nos efforts sur la recherche des documents supportant cette méthodologie ont été vains. C'est pour cette raison que nous ne l'avons pas étudiée dans le chapitre 7 de ce mémoire. Cependant, d'après [CAR99], nous pouvons affirmer ce qui suit.

Burmeister [BUR99], l'auteur de AOAD définit trois modèles pour l'analyse d'un système agents :

- le modèle d'agent : il définit les agents internes au système et leur structure (croyance, plans, objectifs, etc.). Les agents et leurs environnements sont identifiés en utilisant une extension des diagrammes de collaboration pour inclure, la croyance, les motivations, les plans et les attributs de coopération;
- le modèle organisationnel : ce modèle décrit les rapports entre agents du système (l'héritage et les rôles dans l'organisation). Cette description utilise la notation OMT;
- le modèle de coopération : il décrit les interactions entre agents.

3.1.6 Multi-Agents Scenario-Based Method (MASB)

Cette méthode [MOU96a] est destinée aux SMA coopératifs et est composée essentiellement de deux phases :

- la phase d'analyse qui se compose des modèles suivants : modèle de scénarios, modèle fonctionnel de rôle, modèle conceptuel de données, modèle d'interactions utilisateur-système;
- la phase de conception est constituée des modèles suivants : architecture SMA, modèle d'agent, modèle d'objet.

3.1.7 Agent-Oriented Methodology for Enterprise Modeling (AOMEM)

Cette méthode [KEN96] propose une combinaison de Object-Oriented Software Engineering (OOSE) [JAC92] des méthodes de modélisation d'entreprises IDEF (Integration Definition for Function modeling) [FIP93] et de CIMOSA (Computer Integrated Manufacturing Open System Architecture) [KOS93]. Elle est constituée des modèles suivants : le modèle fonctionnel, le modèle des cas d'utilisation, le modèle dynamique et le système agent.

3.2 Les méthodes constituant une extension des méthodes à base de connaissance

Les méthodes à base de connaissance fournissent des techniques pouvant prendre en compte l'état mental des agents. De plus, ces méthodes possèdent une librairie d'outils pouvant être utilisés. Cependant, ces méthodes ne peuvent pas modéliser le comportement social des agents dans un SMA.

Plusieurs méthodes pour la modélisation des SMA, étendant la méthode CommonKADS, [SCH94] ont été proposées (CommonKADS étant la méthode standard de modélisation des connaissances en Europe).

3.2.1 La méthode CoMoMAS

Un agent est vu ici comme une entité ayant des compétences réactives, cognitives coopératives et/ou sociales. La méthodologie CoMoMAS [NOR96] utilise la syntaxe de CML (Conceptual Modeling Language) [SCH94a] pour décrire ses différents modèles qui sont : le modèle d'agent, le modèle d'expertise, le modèle de tâche, le modèle de coopération, le modèle du système et le modèle de conception (figure 3.4).

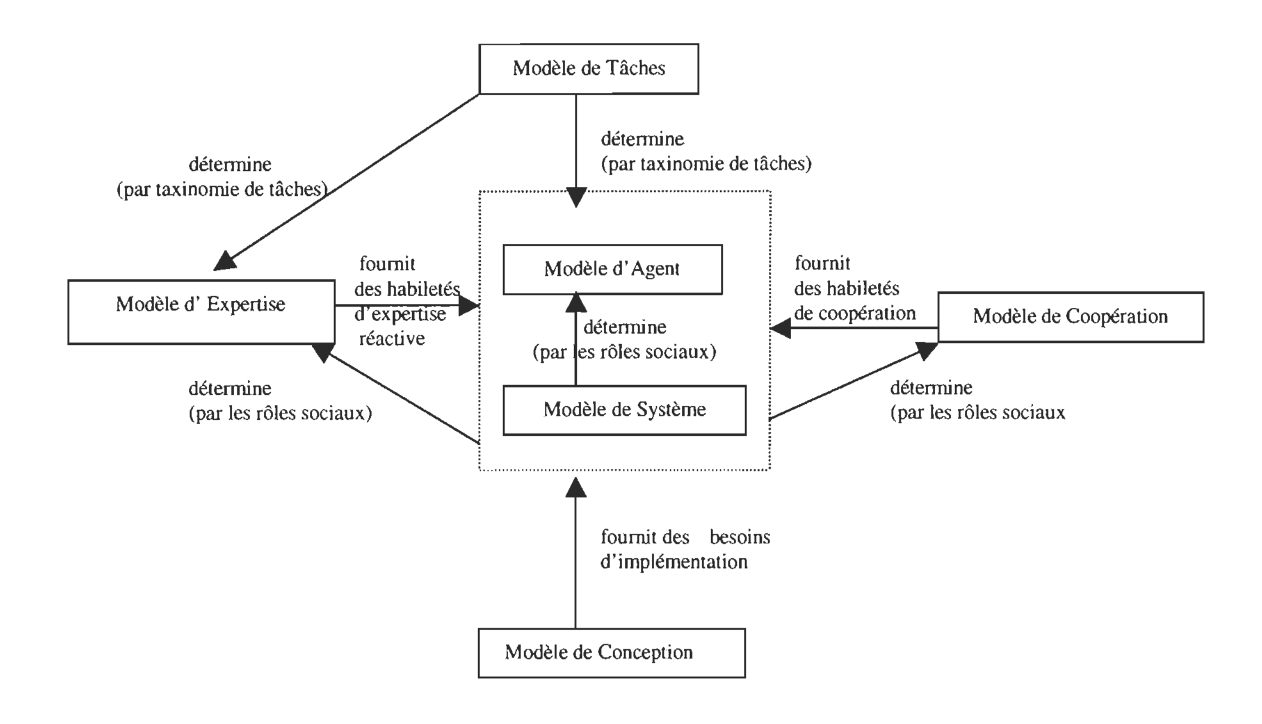


Figure 3.4 : Les modèles de CoMoMAS [NOR96]

3.2.2 La méthode MAS-CommonKADS

Cette méthode [CAR98] est une combinaison de CommonKADS, OOSE (Object Oriented Software Engineering) [JAC92], OMT (Object Modeling Technique) [RUM91] et des protocoles d'agents SDL (Specification and Description Language) [ITU94] et MSC96 (Message Sequence Charts) [EKK96]. Les différents modèles utilisés dans cette méthode sont : modèle agent, modèle des tâches, modèle d'expertise, modèle de coordination, modèle organisationnel, modèle de communication et modèle de conception.

3.3 Les méthodes qui ont été conçues pour un contexte particulier

3.3.1 La méthode Cassiopée

Cette méthode [COL96], dédiée à la robotique collective, distingue trois principales étapes dans la conception d'un SMA :

- les comportements élémentaires des agents sont listés en utilisant les techniques orientées-objet;

- les dépendances entre les agents sont étudiées en utilisant un graphe de couplage. Un graphe de couplage contient les dépendances entre rôles (des agents) qui sont jugées pertinentes pour l'accomplissement collectif d'une tâche. Les chemins et les circuits élémentaires de ce graphe définissent les possibilités de regroupement des différents rôles du domaine et fournissent ainsi une représentation globale de la structure des organisations auxquelles les agents, en jouant ces rôles, peuvent appartenir [COL96];
- Les dynamiques de l'organisation sont décrites en analysant le graphe de couplage.

3.3.2 Cooperative Information Agents Design

Selon [CAR99], cette méthode [EGO97] est destinée aux processus de commerce et est composée des modèles suivants :

- Modèle d'autorisation : décrit les communications autorisées, les obligations entre l'organisation et l'environnement, et les communications internes utilisant les diagrammes d'autorisation;
- Modèle de communication : raffine le modèle précédent en décrivant les détails des contrats entre les agents, et en utilisant les réseaux de Pétri. Les transactions entre les agents sont modélisées en utilisant les diagrammes de transaction qui décrivent les rapports entre les actes de discours et les objectifs;
- Modèle de l'univers de discours : concerne la modélisation des contenus des messages échangés entre les agents; cette modélisation utilise les techniques orientées-objet.

Les informations que nous possédons sur cette méthode ne nous ont pas permis d'en faire une analyse détaillée.

3.4 Conclusion

Dans ce chapitre, nous avons décrit brièvement les méthodologies SMA que nous avons pu recenser dans la littérature. Elles sont classées en deux catégories : soit elles constituent une extension des méthodologies orientées-objet, soit une extension des méthodologies à base de connaissance. Ces méthodologies ont fait l'objet d'une étude approfondie dans le chapitre 7. Il y

en a d'autres (elles n'ont pas été prises en compte par le chapitre 7) sur lesquelles nous n'avons pas eu assez d'information.

Ces méthodologies, dans la plupart des cas, n'ont pas été appliquées à des cas pratiques dans l'industrie ou les entreprises.

Chapitre 4

Quelques outils supportant les méthodologies SMA

Nous allons dans ce chapitre, présenter quelques outils supportant les méthodologies que nous avons abordées dans le chapitre 3. Notre souci n'est pas de donner une description détaillée de ces outils mais de dire qu'ils existent, quitte au lecteur intéressé à aller fouiller la documentation les supportant. Dans la suite de ce document, nous allons utiliser ces outils.

4.1 Les Uses Case Maps

Un Uses Case Maps (UCM) est, à un niveau élevé, une technique de modélisation de scénarios définie pour la conception des systèmes concurrents et temps réel. Un scénario dans le contexte des UCMs est une séquence de responsabilités (événements et activités) internes ou externes qui sont reliées de façon causale dans le but d'offrir une certaine fonctionnalité.

Les UCMs sont utilisés pour décrire et intégrer les cas d'utilisation qui représentent les exigences d'un système. Ils [EAL99] sont des entités structurelles précises qui contiennent beaucoup d'informations dans une forme hautement condensée, rendant capable une personne de visualiser le comportement d'un système. Ils font abstraction des détails de conception et d'implémentation. Il est à noter que la représentation formelle textuelle existe. Cette représentation, basée sur eXtended Markup Language (XML) [CON98] permet aux outils de générer les UCMs ou de les utiliser pour un futur processus et d'analyser les fonctionnalités d'un système [AMY99b]. Il existe actuellement un outil qui supporte la notation UCM et le format XML : le navigateur UCM (UCMNav) [MIG98]. La notation UCM est principalement composée des chemins et des composantes. Un chemin est une séquence de scénarios. Le point d'entrée sur le chemin (schématisé par un cercle noir) représente les préconditions ou le déclencheur du premier scénario. Le point de sortie du chemin (schématisé par une barre verticale) représente les postconditions ou les résultats. Les responsabilités (schématisés par des croix) sont liées de façon causale sur le chemin et représentent des actions, des tâches ou des fonctions à exécuter. On peut combiner plusieurs chemins de plusieurs façons. Par exemple, le point de sortie d'un chemin peut être le point d'entrée d'un autre chemin.

Il existe des constructeurs (ou connecteurs) de chemins qui permettent de représenter les chevauchements. La cardinalité est représentée par N : M. Les composantes sont des entités qui composent le système. Elles peuvent être de différents types selon la nature du système. Les parallélogrammes représentent les composantes actives (processus) et les rectangles les composantes passives (les objets), les rectangles épais représentent les agents. Plusieurs chemins peuvent traverser une composante. Lorsqu'une responsabilité est à l'intérieur d'une composante, cela signifie que cette composante est en charge de l'exécution de la tâche, de la fonction que représente cette responsabilité. La figure 4.1 donne un exemple des Uses Case Maps.

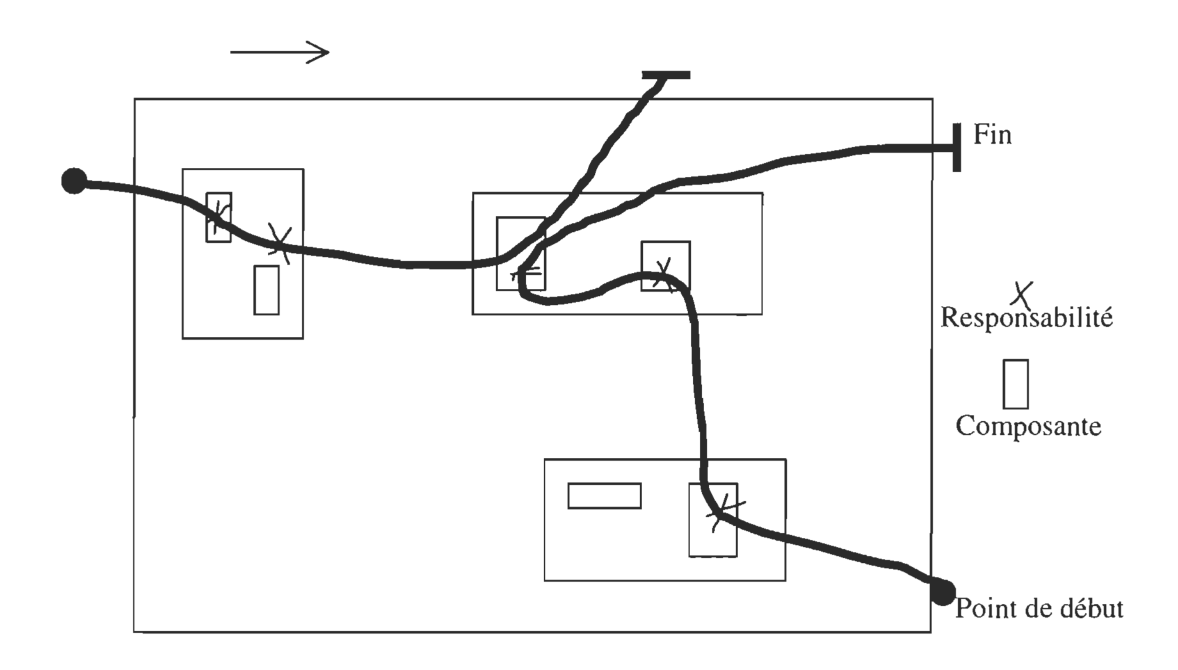


Figure 4.1 : Un exemple des Use Case Maps

Les UCMs peuvent être dérivés des exigences informelles de l'utilisateur. Les UCMs développés à l'université de Carleton par le professeur Buhr et son équipe, depuis 1992, visent à bâtir un pont entre les exigences informelles et la première conception formelle du système. Ils permettent le prototypage rapide du système et la génération des tests fonctionnels [AMY99a].

Pour les SMA, les chemins des UCMs impliquent le comportement coopératif de multiples agents. Les agents collaborent pour réaliser les séquences du chemin.

L'apport de la notation des UCMs pour les systèmes agents est qu'elle permet non seulement de visualiser le comportement global de ces systèmes, mais permet aussi aux experts d'imaginer mentalement les détails d'implémentation.

Ce qui différencie les UCMs des autres notations (OMT, Booch, etc.) est son habileté à condenser plusieurs éléments dans une forme visuelle compacte qui peut rendre facile la communication entre experts et non experts [BUR98a]. Les UCMs fournissent une réduction significative, par rapport aux autres techniques, du nombre de diagrammes nécessaires pour la modélisation des systèmes.

Selon [BUR98a], les UCMs sont particulièrement convenables (au point de vue représentation) pour les SMA, à cause de leur habileté à exprimer dans une notation à un niveau élevé le « quoi » et le « comment » du comportement coopératif et du dynamisme du système.

4.2 Conceptual Modeling Language

Conceptual Modeling Language (CML) [SCH94a] est un langage semi-formel qui permet de décrire le contenu d'un modèle ou d'une application en général. Par exemple, dans la méthode CoMoMAS une application est représentée par un modèle d'application qui est, dans la syntaxe de CML :

```
Application : := APPLICATION Application-name ;
             FUNCTIONAL-VIEW : task-model
             COMPETENCE-VIEW : expertise-model
             COOPERATIVE-VIEW : cooperation-model
             SOCIAL-VIEW : system-model
             REQUIREMENT-VIEW : design-model
             AGENTS : agent-model
          END APPLICATION [ Application-name; ].
```

Cela signifie qu'une application dans la méthode CoMoMAS est décrite par :

- une vue fonctionnelle traduite dans le modèle de tâches;
- une vue des compétences traduites dans le modèle d'expertise;
- une vue coopérative traduite dans le modèle de coopération;
- une vue sociale traduite dans le modèle de conception;
- un niveau agent traduit dans le modèle d'agents.

Nous allons utiliser la syntaxe de CML dans l'annexe pour décrire le cadre CaMuCCoSMA.

4.3 Agent Modeling Language

Agent Modeling Language (AgML) est un langage graphique qui permet de décrire les types d'agents dans un système et leurs interfaces avec d'autres agents. Ce langage est utilisé dans la méthodologie MaSE. AgML utilise des diagrammes pour définir les grandes caractéristiques des SMA. Ces diagrammes permettent de définir, entre autres, le modèle du domaine utilisé pour développer le système. Ce modèle décrit comment les différents types d'agents sont relatés et comment les agents peuvent être combinés pour former le SMA. Une classe d'agents ressemble à une classe d'objets. Cependant, une classe d'agents est définie en terme de services et d'objectifs; alors qu'une classe d'objets est définie en terme d'attributs et de méthodes. Dans le paradigme objet, les rapports entre classes d'objets sont traduits par les associations, alors que dans un diagramme d'agents, ces associations sont des conversations entre agents (figure 4.2). La notion de cardinalité est permise.

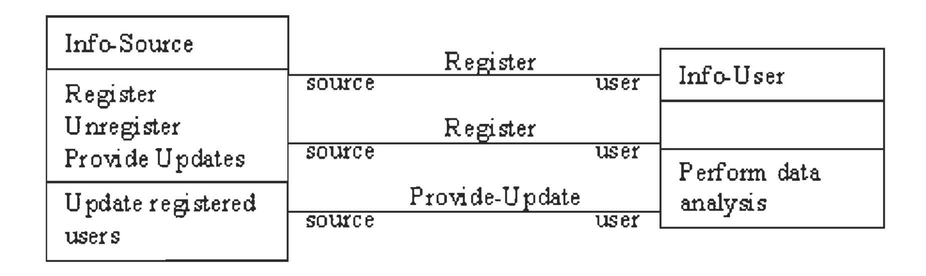


Figure 4.2 : Conversations entre Agents [SCO99]

D'après [SCO99], la syntaxe et la sémantique de AgML sont formelles (basées sur des outils algébriques) et permettent plus facilement le passage de la spécification au code. Elles permettent aussi une réutilisation automatique des composantes et facilitent la vérification. C'est le point de vérification que nous trouvons intéressant et que nous suggérons dans le cahier des charges au chapitre 8 de ce document.

Un autre langage, AgDL, est utilisé dans la méthode MaSE pour la documentation sur l'architecture de chaque type d'agent du système concerné. AgDL, pour Agent Definition Language, est basé sur la logique des prédicats du premier ordre et est utilisé pour compléter la description des comportements internes de chaque agent.

4.4 Quelques autres outils

Il y a d'autres outils qui supportent les méthodologies que nous avons abordées au chapitre précédent. Nous allons nous contenter de les citer ici simplement. Le lecteur intéressé par ces outils pourra consulter leur documentation.

Il s'agit de :

- SMAUL2 : Systèmes Multi-Agents Université Laval [MOU96a]; c'est un environnement qui permet de générer les modules de la méthode MASB [MOU96a];
- MSC : Message Sequence Charts [EKK96];
- SDL : Specification and Description Language [ITU94];
- HMSC : High level Message Sequence Charts [ITU94] qui permet de définir des protocoles de coopération;
- RDD : Responsability Driven Design [WIR90];
- FUSION [COL94] : c'est une notation formelle utilisée par la méthode GAIA pour définir son modèle de rôle.

Il y a d'autres outils qui sont utilisés dans la conception des SMA tels que : DESIRE (framework for Design and Specification of Interacting Reasoning Components) [BRA97] et CoLa [EGO01] qui sont des langages formels de spécification pour les SMA basée sur une décomposition de tâches.

4.5 Un modèle d'interaction dynamique

Le Modèle d'Interaction Dynamique (MID) a été élaboré par Ribeiro A. M. dans sa thèse de doctorat [RIB00] à l'université Joseph Fourier-Grenoble1.

Le MID est un modèle d'interaction où le mécanisme d'interaction joue un rôle dynamique. En effet, une partie de la gestion de l'interaction est retirée de l'agent et transférée vers le mécanisme d'interaction, permettant à l'agent de se concentrer sur le sujet de l'interaction et de laisser de côté les questions qui portent sur comment et avec qui interagir. Le besoin de la conception de ce modèle provient de l'observation de la communication humaine où les messages peuvent éventuellement être transmis de façon indirecte avec l'aide d'un messager. Le messager utilise l'information fournie par l'émetteur plus ses propres connaissances pour délivrer le message [RIB00]. Dans le MID, un mécanisme d'interaction est l'ensemble des services et des messages actifs. Pour le bon fonctionnement du MID, la plate-forme (figure 4.3) sur laquelle les agents sont connectés doit fournir quelques services de base, qui sont :

- un Gestionnaire des Agents (GA) qui contrôle les agents connectés au système, leur identité et leur état. C'est un service de «Pages Blanches»;
- un Service d'Organisation du Système (SOS), qui doit connaître l'organisation des agents et être capable de répondre à des questions sur ce sujet;
- un Service de Répertoire (SR), qui consiste en un catalogue des services (capacités) disponibles dans le système et des besoins déclarés par les agents. C'est un service de «Pages Jaunes»;
- un Service d'Interaction (SI), qui doit fournir le support pour la création et le fonctionnement des messages actifs.

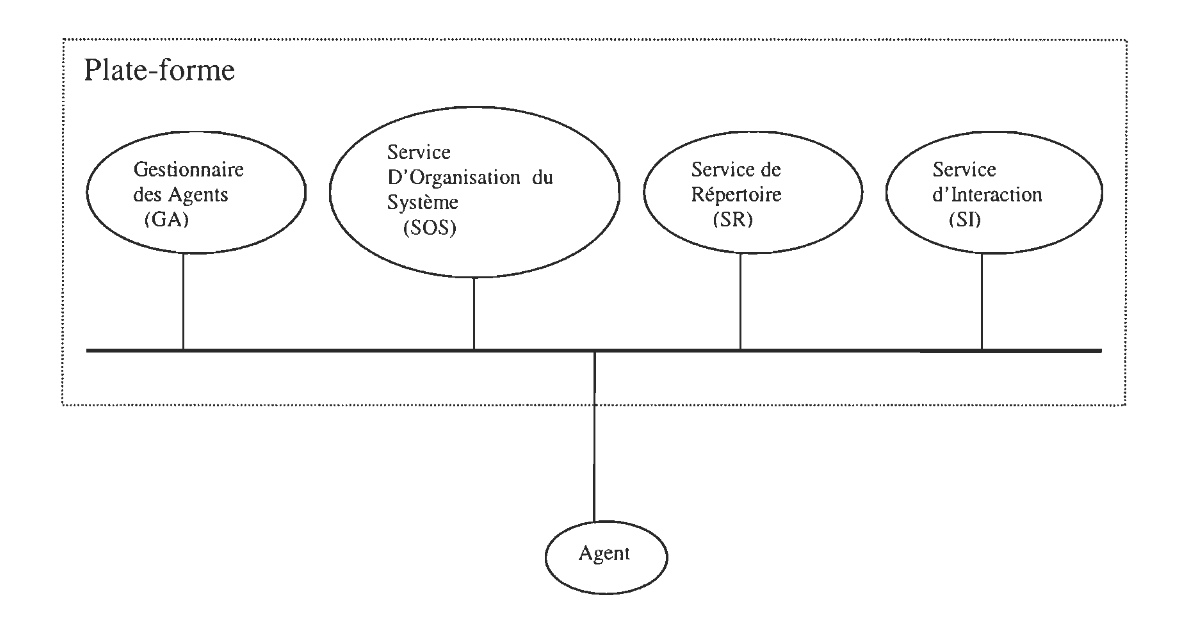


Figure 4.3 : Plate-forme générique pour le MID [RIB00 : p. 40]

Les Messages Actifs

Les Messages Actifs sont les principaux composants du MID. Les messages actifs jouent un rôle central dans l'interaction. Ils reçoivent la responsabilité de mener à terme une interaction. Un message actif peut être regardé comme étant un agent spécialisé dans la livraison de messages. Le message actif interagit à deux niveaux différents : avec son émetteur (avec qui les échanges sont au niveau des états mentaux) et avec les autres agents du système (figure 4.4).

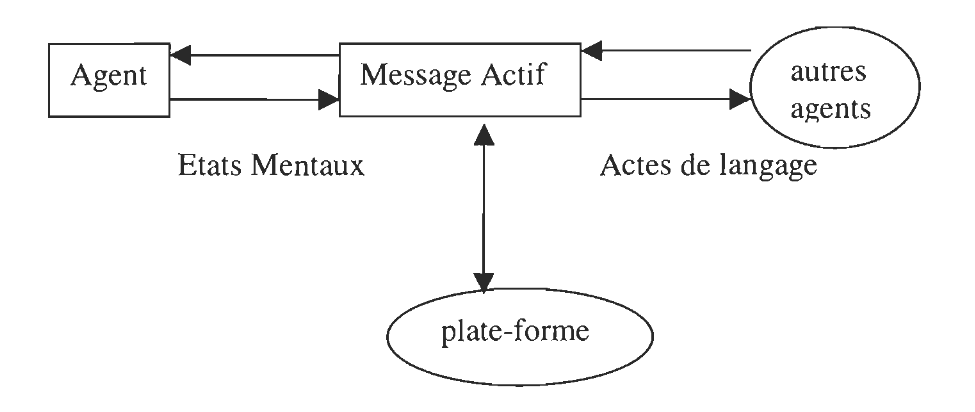


Figure 4.4 : Interfaces d'interaction d'un message actif [RIB00 : p. 41]

Le MID permet l'hétérogénéité pour l'interaction entre agents. La gestion et le suivi des protocoles d'interaction sont complètement délégués au mécanisme d'interaction.

4.6 Conclusion

Nous avons décrit brièvement quelques outils supportant les méthodologies SMA. Nous allons utiliser les Uses Case Maps, le langage Agent Modeling Language (AgML) et le Modèle d'Interaction Dynamique, dans les suggestions au chapitre 8. Le langage Conceptual Modeling Language (CML) sera utilisé pour décrire dans l'annexe le cadre CaMuCCoSMA. La raison fondamentale pour laquelle nous avons fait la description de ces outils est que nous les avons utilisés dans les chapitres suivants. Cela permettra au lecteur d'en avoir une idée. Il faut signaler que ces outils sont des outils de spécification, de modélisation et de conception. Ils ont été seulement appliqués aux études de cas théoriques décrits dans les documentations et n'ont pas été appliqués aux applications réelles industrielles.

Chapitre 5

Quelques plateformes SMA

Bien que les recherches sur les agents aient été commencées depuis plus de deux décennies, peu d'efforts ont été dirigés vers un consensus d'architecture SMA. Une des raisons possibles (de ce manque de consensus) pourrait être l'absence d'une conception commune, dans le cercle de la recherche, des premiers principes généraux sur lesquels doivent se baser les architectures SMA [RIB00]. C'est tout récemment que plusieurs groupes de chercheurs et d'industriels indépendants ont commencé à poursuivre la standardisation de la technologie multi-agents. Parmi ces groupes on peut citer :

- Object Manager Group (OMG);
- Foundation for Intelligent Physical Agents (FIPA);
- Knowledgeable Agent-oriented System (KAOS);
- General Magic Group.

5.1 Le modèle de OMG

Le groupe OMG propose un modèle de référence comme une ligne directrice pour le développement des technologies agents [YIR95]. Ce modèle fait ressortir les caractéristiques d'un environnement agent composé d'agents (composantes) et d'agences (places ou lieux de rencontres), qui sont des entités devant collaborer en utilisant des modèles et des politiques généraux d'interactions. Dans ce modèle, les agents sont caractérisés par leur compétence (de déduction, de planification, etc.), leurs types d'interaction (synchrone, asynchrone), et leur mobilité (statique, mobile avec ou sans état). D'autre part, les agences supportent l'exécution concurrente d'agents, la sécurité et la mobilité des agents, entre autres [ROB99].

5.2 Le modèle de FIPA

FIPA est un groupe multidisciplinaire poursuivant la standardisation de la technologie agent. Ce groupe a mis sur pied une série de spécifications pour diriger le développement des

SMA. Les plus importantes sont la spécification de la gestion d'agent (Agent Managment) [FIP97a] et la spécification de langage de communication agent (ACL) [FIP97b]. Le langage FIPA-ACL suit le style de KQML (utilisant des performatifs issus de la théorie des actes de langage et quelques paramètres complémentaires), mais avec une sémantique mieux spécifiée. Le langage prévoit aussi l'utilisation de protocoles d'interaction. Il existe une bibliothèque de protocoles standards [RIB00].

L'approche de FIPA pour le développement des SMA est basée sur un paradigme minimal pour la gestion des agents dans un environnement ouvert. Ce paradigme est décrit en utilisant un modèle référentiel (qui spécifie un environnement normatif dans lequel les agents existent et opèrent) et une plate-forme agent (qui spécifie une infrastructure pour le déploiement et l'interaction des agents).

5.3 Le modèle de KAOS

L'architecture proposée par ce groupe, pour les SMA, est une architecture distribuée et ouverte pour les agents logiciels [KAO01]. Elle décrit les implémentations des agents (partant de la notion de simple agent à la notion de rôle d'agent, tel que les médiateurs et les apparieurs) et élabore, sur la base des interactions dynamiques d'agent à agent, des messages de communication en utilisant des politiques de conversation.

5.4 Le modèle de General Magic

General Magic est une tentative de recherche commerciale sur la technologie d'agents mobile pour le commerce électronique. Conceptuellement, cette technologie modélise un SMA comme un marché électronique qui permet aux fournisseurs et aux consommateurs de se rencontrer pour faire des transactions d'affaires. Ce marché est modélisé comme un réseau d'ordinateurs supportant une collection de lieux qui offrent des services aux agents mobiles. Les agents mobiles, qui sont représentés comme des entités qui résident dans un endroit particulier en un temps donné, sont décrits suivant les compétences suivantes [WHI97] :

- ils peuvent voyager : ils ont le droit de se déplacer d'une place à une autre;

- ils peuvent rencontrer d'autres agents : une rencontre permet aux agents à une même place d'invoquer des procédures d'autres agents.
- ils peuvent créer des connexions qui permettent à un agent de communiquer avec un autre agent d'une place différente.
- ils ont une représentativité : qui indique le monde physique individuel ou l'organisation que l'agent représente;
- ils possèdent des permis qui indiquent les compétences des agents.

5.5 Autres plateformes SMA

Les informations que nous donnons ici sont une synthèse de celles tirées de [GUE01].

5.5.1 CORMAS

CORMAS est une plate-forme qui a été créée au Cirad, qui est un institut de recherche en agronomie pour le développement, au sein de l'équipe GREEN (Gestion des ressources renouvelables, environnement) qui concevait la gestion des ressources comme un problème d'interactions entre des dynamiques naturelles et des dynamiques sociales [COR01]. CORMAS est un environnement de développement de SMA pour des problèmes de dynamique et d'usage de ressources. C'est un environnement qui propose à l'utilisateur de se construire son modèle (en s'appuyant éventuellement sur des modèles préexistants) pour le traiter par simulation. CORMAS propose, selon l'application, des agents très réactifs ou délibératifs spatialisés (situés sur l'espace), communicants, socialement contrôlés ou pas, raisonnant sur le temps ou pas, stratèges ou opérationnels. CORMAS s'appuie sur le langage Smalltalk pour développer ses modèles. CORMAS fournit des bibliothèques d'agents, d'environnement et de protocoles. CORMAS est destiné aux résolutions de problèmes, aux simulations et à l'intégration.

5.5.2 DIMA

DIMA (Développement et Implémentation de Système Multi-Agents) est un environnement de développement de SMA qui a été créé dans le cadre de la thèse de Z. Guessoum [DIM01]. DIMA propose des agents réactifs, cognitifs, hybrides, autonomes,

adaptables. DIMA utilise les langages Smalltalk-80 et JAVA pour implémenter ses modèles. Il fournit des librairies offrant les briques de base pour construire des modèles d'agents divers. Les langages de communication entre agents sont basés sur les langages KQML et ACL de la FIPA. Il est destiné aux simulations et aux résolutions de problèmes (éventuellement temps réel).

5.5.3 GEAMAS

GEAMAS (GEneric Architecture for Multi-Agents Simulations) est une plate-forme logicielle générique pour la modélisation et la simulation multi-agents entièrement implémentée en JAVA. Elle a été développée à l'Université de la Réunion par l'équipe MAS2. L'architecture logicielle de GEAMAS s'appuie sur un micro-noyau générique, JAAFAAR [CAL99], offrant les structures et mécanismes minimaux nécessaires à l'implémentation de SMA, autour duquel gravitent un certain nombre d'extensions logicielles spécialisées (modules d'apprentissage, d'auto-organisation, de conception assistée, etc.).

5.5.4 MAGIQUE

MAGIQUE est une plate-forme générique de développement de SMA, développée en JAVA (JDK 1.x) et destinée aux résolutions de problèmes distribués [MAG01]. MAGIQUE est une architecture distribuée sur réseau hétérogène et dont la structure de contrôle est hiérarchique. Elle utilise des agents réactifs et pro-actifs.

5.5.5 MASK

MASK (Multi-Agent System Kernel) est un environnement de développement de SMA au sein de l'équipe MAGMA du laboratoire LEIBNIZ/IMAG/CNRS de Grenoble [CNR01]. MASK est fondé sur l'approche d'Analyse/Conception Voyelle (AEIO) et constitue le support logiciel de cette méthode. MASK fournit des librairies d'agents hybrides (ASTRO) et réactifs (PAGORG), d'interactions (IL et PACO) et d'organisations statiques (RESO) et dynamiques (MID). MASK utilise les langages de programmation C, C++ et JAVA pour développer ses modèles. Il est destiné à l'intégration, aux simulations et aux résolutions de problèmes.

5.5.6 MAST

MAST (Multi Agent System ToolKit) est un environnement de développement de SMA qui a été développé par l'équipe SMA à l'École Nationale Supérieure des Mines de Saint-Etienne [MAS01]. MAST fournit plusieurs infrastructures d'agents, d'environnement, d'interaction et d'organisation. Il utilise les langages C++, JAVA, Jess, Clips ou Prolog pour implémenter ses modèles. Il permet l'hétérogénéité des agents. Il est destiné à l'intégration, aux simulations et aux résolutions de problèmes.

5.5.7 MERCURE

MERCURE est une plate-forme générique en cours de développement dans le cadre d'une collaboration entre le LIMSI-CNRS (groupe Langage et Cognition), la société AEGIS et le LGIS, au sein du projet EUREKA PVS98. Elle se compose des agents techniques spécifiés par la FIPA (Directory Facilitator, Agents Management Service, Agent Communication Channel, Agent Naming Service) et d'agents spécialisés que l'environnement permet de développer et de faire fonctionner. Le langage de communication entre agents est ACL (Agent Communication Language). Un langage de programmation LPPA (Langage de Programmation des Plants d'Agents) a été développé et permet aux agents de déclarer leurs compétences auprès des facilitateurs. Les agents sont développés en Smalltalk, mais peuvent utiliser des modules logiciels écrits dans d'autres langages, en s'interfaçant avec CORBA, COM ou C.

5.5.8 MoCAH

MoCAH pour Modélisation de la Coopération entre Agents Hétérogènes est une plate-forme de développement de SMA développée au département d'informatique à l'Université Paris 8 en France [MOC01]. MoCAH a été implémentée en Smalltalk-80 et utilise des agents hétérogènes qui coopèrent pour résoudre un problème commun. La résolution coopérative s'organise autour d'un agent privilégié que les auteurs de MoCAH ont appelé agent pivot.

5.5.9 OSACA

La plate-forme OSACA (Open System of A Synchronous Cognitive Agents) a été développée par Edson Scalabrin et Jean-Paul Barthhès à l'UTC (Université de Technologie de

Compiègne) [OSA01]. Le type d'application visé est celui faisant intervenir un petit nombre d'agents cognitifs complexes. Les agents sont développés en LISP, C ou C++. Les agents peuvent quitter ou intégrer dynamiquement le système. OSACA peut utiliser, entre autres, des agents interfaces qui assistent l'utilisateur dans l'exécution de ses tâches.

5.6 Conclusion

L'objectif de ce chapitre était de donner une brève description des plateformes. Il y a peu d'efforts sur la standardisation des plateformes SMA. Les travaux de Ricordel [RIC00] prouvent bien ce manque de standardisation. Ce sont seulement quelques groupes (OMG, FIPA, KAOS, General Magic Group) de recherche et d'industriels qui ont commencé à poser des jalons de cette standardisation. Nous espérons que notre travail de recherche va contribuer à la poursuite de cette standardisation. Nous pensons que l'unification des méthodologies pourra sans doute influencer fortement cette standardisation de plateformes SMA.

Troisième partie

Proposition d'un cadre pour la comparaison des méthodologies SMA

Chapitre 6

Le Cadre multidimensionnel de critères pour la comparaison des méthodologies SMA (CaMuCCoSMA)

6.1 Problématique

Bien qu'il y ait actuellement un regain pour les techniques et les méthodologies de modélisation et de conception des SMA, le développement de ces derniers engendre d'énormes problèmes et reste donc un domaine ouvert. Pour que la technologie orientée-agent puisse connaître un véritable succès, il faut une méthodologie rigoureuse dans les différentes phases du processus de développement de systèmes informatiques multi-agents [WOO98].

La proposition du cadre CaMuCCoSMA est une approche de solution. En effet, ce cadre nous a permis d'évaluer les principales méthodologies de développement des SMA (qui existent) puis de dégager leurs points communs, leurs insuffisances et leurs avantages. Ce qui nous a permis de jeter un regard objectif sur les perspectives d'avenir.

D. Pascot et C. Bernadas [PAS93] avaient proposé en 1993 un cadre de référence leur permettant de comparer des méthodes de conception de systèmes d'information informatisés. Ce cadre de référence a été repris par les auteurs de [ADA99a] et adapté pour répondre à leur problématique qui était l'analyse et la conception de systèmes interactifs dédiés aux systèmes administratifs complexes. Les critères du cadre CaMuCCoSMA, que nous proposons ici, ont été fortement inspirés de ceux de [ADA99a], des critères définis pour l'étude comparative des plateformes SMA du groupe AFIA/PRC-13 SMA [AFI98], et adaptés au développement des SMA. Nous avons aussi tenu compte de [ROB99]. Nous avons pris le soin de définir clairement tous les termes pertinents de ce cadre, y compris ceux importés. L'originalité de ce cadre est l'importation et l'adaptation des concepts de l'orienté-objet et du génie logiciel au développement de méthodologies SMA. Nous avons définit de nouveaux critères et nous avons donner une sémantique à ceux que nous avons empruntés des autres travaux cités ci-dessus.

Notre souci est de rendre plus systématique la conception des SMA. Aussi, nous voudrions contribuer à la standardisation des méthodologies et des plateformes SMA.

6.2 Le cadre CaMuCCoSMA

Le cadre CaMuCCoSMA est constitué de six dimensions. Les auteurs de [ADA99a] ont ajouté aux quatre dimensions (Méthodologie, Représentation, Organisation, et Technologie) de [PAS93] la dimension Coopération. À ces cinq dimensions, nous avons ajouté une sixième intitulée dimension Agent. Chaque dimension est définie avec un certain nombre de critères. Ces critères permettent de comparer les méthodologies SMA. C'est une sorte de repère dans lequel on se placera pour comparer les méthodologies SMA. L'outil que constitue ce cadre va permettre de déboucher sur un cahier de charges pour la conception d'une méthodologie SMA.

Nous définissons, dans cette section, les critères pour l'évaluation et la comparaison des méthodes de développement de SMA. Nous avons donné la signification de chacune de ces valeurs.

Les lignes ou parties de lignes qui sont en italique indiquent les informations que nous avons tirées des documents [ADA99a], [AFI98], [ROB99] pour concevoir le cadre CaMuCCoSMA.

6.2.1 Dimension Méthodologie

La description du cadre méthodologique d'une méthode est très importante pour la conception des SMA. Cette description, qui fait défaut ou qui est plus ou moins voilée dans les méthodes de conception des SMA, permettra aux concepteurs d'assurer leur fonction selon une optique bien précise. Cette dimension décrit le processus de développement suivi par la méthodologie, le modèle qui supporte ce processus et l'approche utilisée. Elle indique la prise en compte du point de vue de l'utilisateur (client) dans la méthodologie. De façon générale, elle décrit l'environnement de développement des SMA auxquels la méthodologie peut s'appliquer.

Aux cinq critères retenus dans [ADA99a], deux critères ont été ajoutés : les critères Disponibilité de support logiciel ou méthodologique (l'existence d'un outil informatisé supportant une méthode la rend plus opérationnelle) , la Réutilisabilité de modèles (c'est un point

qui rend une méthode générique); ces deux critères ont été tirés de [AFI98]. Ce qui fait au total sept critères pour cette dimension, qui sont :

1-*Étapes du processus* Ce sont les étapes classiques du processus de développement du Génie Logiciel. Ce critère permet d'identifier les différentes phases qui existent dans le processus de développement de la méthode. Dans [ADA99a], ce critère est intitulé Étapes concernées, mais nous avons préféré Étapes du processus pour être plus précis.

- *analyse :* ensemble d'activités dont la finalité réside dans l'expression du besoin ou du problème. Étude du domaine d'application (phase exploratoire).
- modélisation : processus permettant de représenter les données issues de l'analyse. Élaboration de modèles.
- spécification : description de ce que doit faire le logiciel en évitant des décisions prématurées de réalisation.
- conception : ensemble d'activités dont la finalité est d'élaborer une solution à partir d'un problème déjà modélisé. Description de l'architecture du logiciel; spécification de ses divers composants; description pour chaque composant de la manière dont ses fonctions sont réalisées.
- validation : compatibilité entre l'interprétation du modèle par l'utilisateur (client) et le point de vue de celui-ci. A-t-on décrit le bon produit?
- vérification : processus qui permet de tester le produit. A-t-on un produit correct?
- évaluation ergonomique : étude de l'interface homme-machine (prise en compte éventuelle de la collaboration entre concepteur et spécialiste de l'ergonomie).

2- *Modèles de développement* Ce sont les modèles du Génie Logiciel sur lesquels se base le cycle de développement d'une méthode. Dans [ADA99a], ce critère se nomme Cycle de développement. Ce sont :

- le modèle cascade : c'est une approche de développement séquentielle dans laquelle il n'y a pas chevauchement des étapes de développement au cours d'un projet.
- le modèle en V : c'est une variante du modèle en cascade, qui montre la symétrie et la complémentarité qui existe entre les phases de production et les différentes phases de test (pré-test et test).

- le modèle en spirale : c'est une approche de développement qui consiste à ajouter à chaque étape du modèle cascade une phase de prototypage et une revue technique (validation/révision) avec les utilisateurs. Cette tâche de validation consiste à vérifier la conformité du modèle aux besoins de l'utilisateur.
- le modèle incrémental (ou évolutif) : c'est une approche de développement qui consiste à faire une extension progressive des fonctionnalités du système à partir de l'étape de la conception détaillée. Le développement par incrément permet le parallélisme.
- le modèle nabla : c'est un modèle qui ressemble au modèle en V, dont l'objectif est de situer les étapes nécessaires au développement de systèmes interactifs, en caractérisant les interfaces hommes-machines.

3- <u>Approche de développement</u> Ce critère permet d'identifier l'approche suivie par la méthode de développement de SMA. Celle-ci peut être :

- descendante (ou top-down) : approche qui supporte la construction par module et qui est fondée sur la décomposition fonctionnelle du problème du général au particulier, et le raffinement graduel de celui-ci.
- ascendante (ou bottom-up) : c'est une approche par assemblage de composants ou de composition. On s'intéresse aux différents composants, puis on remonte progressivement par composition.
- évolutive : on démarre par les sous-ensembles de composants, ensuite on peut soit descendre au niveau bas, soit remonter au niveau supérieur.

4- <u>*Degré d'implication de l'utilisateur*</u> Les concepteurs de logiciels ont parfois tendance à ignorer le point de vue de l'utilisateur. Or, ce point de vue permettra d'éviter les erreurs de sémantique dans la spécification des besoins de l'utilisateur. Ainsi, le coût du logiciel sera moins élevé. Ce critère indique si la méthode fournit des moyens pour la communication entre concepteurs et utilisateurs. Nous avons modifié les valeurs de ce critère, car nous considérons que, dans toute méthodologie de conception de systèmes informatiques, l'utilisateur intervient, tout au moins pour livrer ses exigences au concepteur. Les valeurs de ce critère sont :

- faible : l'utilisateur intervient seulement au début et à la fin du projet.

- moyen : l'utilisateur intervient au milieu, mais pas dans toutes les phases du développement du système.
- forte : la présence de l'utilisateur se fait sentir tout au long du développement du système auquel la méthodologie peut s'appliquer.

5- *Moment d'implication de l'utilisateur*

- début : intervient tout juste avant la phase d'analyse pour livrer ses besoins.
- milieu : intervient pendant les phases d'analyse, de spécification et de conception.
- fin : après la conception pour prendre le produit final.

6-*Réutilisabilité de modèles* Ce critère indique si la méthode fournit ou permet de fournir une librairie de modèles réutilisables. La fourniture de modèles réutilisables par une méthode la rend générique, ce qui évitera des pertes de temps. Il est même à signaler qu'il existe des axes de recherche sur l'utilisation des patterns pour l'analyse et la conception des SMA ([SAU00] [KEN96]).

7-*Disponibilité de support logiciel* ou méthodologique Ce critère indique s'il existe des outils qui supportent la méthode (existence de bibliothèques d'agents, de composantes d'agents, d'organisations, d'interactions, d'environnement, de supports techniques ou méthodologiques). L'existence d'un outil qui supporte une méthode la rend plus opérationnelle.

6.2.2 Dimension Représentation

Après l'analyse des besoins, devrait suivre une représentation ou un modèle qui traduira d'abord la vision globale du système du point de vue externe. Cette représentation, comme le font les ingénieurs et les architectes, sera d'abord visuelle et graphique pour en faciliter la compréhension, l'interprétation et la communication. La représentation est une étape nécessaire dans le développement de systèmes et, en particulier, les systèmes multi-agents. Toute méthode de conception de systèmes informatiques cherche à représenter, dans un formalisme donné, les résultats issus de l'analyse. Ce formalisme, qui correspondra à la photocopie du système étudié, devrait être simple et clair pour faciliter la compréhension, l'interprétation, et la communication entre experts, d'une part, et experts-utilisateurs, d'autres part. La maîtrise de la complexité d'un système exige des moyens d'assemblage adéquats dans l'analyse et la représentation. Les critères

"Découpage du système" et "Formalisme" de [ADA99a] ont été retenus dans notre cadre. Nous avons ajouté à ces deux critères deux autres critères, "Qualité des modèles" (qui pourrait constituer une métrique pour mesurer la complexité d'une méthodologie) et "Séquencement" (l'ordre de dérivation des modèles).

Cette dimension décrit les principes et les formalismes utilisés à l'étape de la modélisation de la méthode. Les critères sont :

1- *<u>Le découpage du système</u> Ce critère indique les moyens d'assemblage des éléments décelés dans l'analyse. Celui-ci est réalisé par* :

- niveaux d'abstraction : séparation en niveaux conceptuels et organisationnels.
- généralisation et spécialisation : création d'une famille de classes et création d'une sous-classe à partir d'une classe.
- type-occurrence : classe-instanciation (on instancie une classe par un objet).
- stratégie-tactique : il s'agit d'étudier les stratégies à long terme (le pourquoi) du système ainsi que les tactiques (le comment).

2- <u>Le formalisme</u> Ce critère décrit les formalismes utilisés (schémas, concepts et règles) par la méthode. On s'intéresse aux formalismes de données, d'activités, de traitements et de la dynamique du système.

3- <u>Séquencement</u> Une méthodologie devrait fournir des directives pour la dérivation de modèles et le lien entre ces modèles. Cela permettrait aux concepteurs de systèmes d'être plus rapides dans leur tâche de conception. Ce critère indique l'ordre de dérivation et le lien entre les modèles de la méthodologie.

4- <u>Qualité des modèles utilisés dans la méthodologie</u>

- nombre de modèles dans la méthodologie.
- cohérence des modèles : pas de redondance. Tout ce qui se rapporte à un même concept se range sous une même entité; ou bien tout ce qui se trouve sous une même entité se rapporte à un même concept.
- complétude des modèles : l'ensemble des modèles produits doit rendre compte de la totalité de l'univers du discours. Est-ce que l'ensemble des modèles couvrent tous les aspects d'un SMA.

- complexité des modèles : inter-relation entre les modèles (couplage); le couplage est bien sûr nécessaire mais il doit cependant être limité au strict minimum (un couplage trop élevé rendra plus difficile la validation et la vérification des modèles); nombre d'éléments à maîtriser pour construire les modèles.

6.2.3 Dimension Agent

Cette dimension est particulièrement importante pour les systèmes composés d'agents. Elle permet de décrire la nature et les caractéristiques propres aux agents disponibles dans le SMA auquel la méthode peut s'appliquer. Cette description ne rentre pas dans les détails techniques de conception des agents mais plutôt fait état des concepts dont on doit tenir compte pour construire ces agents. La performance, l'efficacité et l'efficience d'un système est fonction de ces caractéristiques. Les attributions des agents constituent un facteur important pour leur comportement social et coopératif. En effet, elles ont pour but de pourvoir les agents d'une capacité d'anticipation et de planification qui leur permet d'optimiser leurs comportements. Elles permettent à un agent de prédire le comportement futur des autres. Quatre critères ont été associés à cette dimension. À part le critère Nature des agents, qui a été tiré de [AFI98] les trois autres ont été tirés de [ROB99]. Nous les avons adapté à notre contexte de recherche. Nous avons ajouté "comportement délibératif" aux attributs des agents. Les critères sont les suivants :

1- *Nature des agents* : Ce critère indique si les agents présents dans les modèles de la méthode sont :

- homogènes : de même nature. Tous les agents sont construits sur le même modèle (exemple : une colonie de fourmis).
- hétérogènes : de nature différente. Les agents sont construits avec des modèles différents, de granularité différente (un écosystème par exemple).

2- *Type d'agents* Ce critère indique les types d'agents que la méthode permet de représenter.

- agents intelligents (ou BDI) : qui sont vus comme des entités qui imitent les processus mentaux ou simulent des comportements rationnels (exemples : les robots footballeurs [COL96]).

- agents interfaces ou personnels : ce sont des entités qui assistent les utilisateurs dans l'exécution d'une tâche (exemple : le système de gestion du trafic aérien dans [KIN96]). Ils fournissent une assistance proactive à l'utilisateur pour une application précise.
- agents mobiles : qui sont des entités capables d'errer dans des environnements en réseau pour accomplir leurs objectifs (exemple : un virus).
- agents d'information : qui sont des agents spécialisés pour filtrer et pour organiser, de façon cohérente les données dispersées et sans lien (exemple : un moniteur de la page Web qui prend en charge les demandes d'informations, effectue les attributions de ressources et contrôle l'enchaînement des tâches [SCO99]).
- agents autonomes : qui sont capables d'accomplir des actions non supervisées (responsables de leurs comportements). Il est capable d'agir sans l'intervention d'un tiers.

3- *Attributs des agents* Ce critère indique les caractéristiques intrinsèques des agents que la méthode utilise. Ces caractéristiques ont certainement une influence sur la performance, l'efficacité et l'efficience du système auquel la méthode peut s'appliquer.

- adaptabilité : habileté à apprendre et à s'améliorer avec l'expérience, acquisition dynamique des connaissances.
- autonomie : objectif non dirigé, comportement proactif et d'auto démarrage (self-starting behaviour). Degré d'autonomie sociale : indépendant, semi-autonome, contrôlé.
- comportement coopératif : habileté à travailler avec d'autres agents pour atteindre un objectif commun. Degré de coopération : coopératif, compétitif, antagoniste.
- capacité déductive : habileté à agir sur des spécifications de tâches abstraites. Un agent peut formuler des hypothèses sur ces tâches et établir des conclusions.
- habileté de communication : habileté à communiquer avec d'autres agents avec un langage qui ressemble beaucoup plus aux actes de discours humains que des symboles de programmes (les actes de discours indiquent les actions intentionnelles effectuées au cours d'une communication).

- mobilité : habileté à émigrer dans une direction désirée, d'une plate-forme à une autre.
- personnalité : habileté à manifester des attitudes d'un caractère humain (un agent peut être égoïste, jaloux, etc.).
- réactivité : habileté à réagir selon les états de son environnement.
- continuité temporelle : persistance de l'identité et de l'état sur de longues périodes.
- comportement délibératif : habileté à décider dans une délibération (autiste, à l'écoute des autres, responsable, manager).

4- *Attributions des agents* Comment prédire le comportement d'un agent, ne connaissant pas sa structure interne? Quelles représentations attribuées à un agent qui ressemblent le mieux à ses comportements observables (externes) et qui créent de façon sûre ses comportements attendus. Les trois stratégies de Dennett [DEN87] essayent de couvrir ces types de représentation (dans lesquelles des entrées vont créer des sorties) basées sur :

- la stratégie physique : étant donné des caractéristiques de l'environnement et de l'agent, on peut espérer qu'il se comporte de plusieurs façons (par exemple une balle de tennis lancée en l'air). Cette stratégie prend en considération l'état physique du système.
- la stratégie de l'entité : les entrées/sorties peuvent être comprises mais pas les productions spécifiques (exemple : on peut voir les effets mortels d'un accident dû à un mauvais freinage d'une voiture à toute vitesse, mais le scénario propre à l'accident peut être difficile à comprendre).
- la stratégie intentionnelle : les prédictions sont basées sur les hypothèses de l'environnement (les agents BDI par exemple). Cette stratégie consiste à attribuer à l'être observé une intention cohérente avec ses actions. Un agent doit pouvoir percevoir les actions des autres agents, en déduire leurs intentions, puis adopter un comportement.

Afin de gérer la dynamique des organisations dans les SMA, il est nécessaire de doter chaque agent d'un module de reconnaissance ou d'attribution d'intention. Reconnaître l'intention d'autrui permet la coordination, guide la coopération et, enfin, assure la pertinence (au sens

d'efficacité) des interactions [PAT98]. La méthodologie étudiée possède-t-elle un modèle d'attribution ou fournit-elle des mécanismes d'attribution? Dans l'affirmative, est-ce que la construction de ce modèle est basée sur une ou plusieurs des stratégies de Dennett ?

6.2.4 Dimension Organisation

Selon Fox [FOX81], on peut définir une organisation comme une structure décrivant comment les membres de l'organisation sont en relation et interagissent afin d'atteindre un but commun : les sorties (output) de l'organisation [FER95]. La structure d'une organisation est fonction de l'environnement dans lequel elle évolue, des ressources disponibles pour produire les sorties, ainsi que de la nature de ces sorties. Cette structure peut être statique, donc conçue à priori par le programmeur du système, ou dynamique, comme on peut la trouver dans les SMA ouverts. La construction d'un SMA nécessite la construction d'une structure organisationnelle sous-jacente. Cela veut dire que le concepteur du système doit, à priori, faire un choix convenable de cette structure afin de maîtriser la complexité du système. La complexité pouvant être vue comme la quantité d'informations que ce système peut traiter [BAE98].

Cette dimension définit la structure que doit avoir le système et les caractéristiques de l'environnement auquel le système est destiné (ou sur lequel les agents agissent). La dimension indique si la méthode précise cette structure et ces caractéristiques. Les trois critères de [ADA99a] ont été retenus dans notre proposition mais leurs valeurs ont été quelque peu modifiées compte tenu de l'architecture et des propriétés des SMA.

1-*Image d'organisation* *C'est la possibilité que la méthode a de traduire ou de représenter les types d'organisation* (un type d'organisation étant défini par un ensemble de rôles et un ensemble de rapports entre ces rôles) cités ci-dessous. Nous avons retenu quatre types de ceux proposés dans [ADA99a], qui sont, à notre avis, les principaux types qu'on remarque dans les SMA.

- *système hiérarchique* : c'est une architecture dans laquelle un niveau supérieur définit les contraintes et les objectifs à atteindre par le niveau suivant.
- *système distribué* : c'est une architecture dans laquelle les agents sont connectés par l'intermédiaire d'un réseau pour atteindre un objectif donné. Ils sont physiquement distribués. On oppose généralement système distribué à système centralisé.

- *système ouvert* : c'est une architecture dans laquelle les agents n'ont pas besoin d'être conceptualisés ensemble pour partager un objectif commun et peuvent dynamiquement quitter ou intégrer le système. Les agents peuvent être hétérogènes. Les rapports inter-agents peuvent changer pendant l'exécution (un café par exemple où les gens rentrent et sortes librement).
- *système holonique* : c'est une architecture dont la structure résulte de la combinaison d'une structure hiérarchique et d'une structure hétérarchique (distribuée).

2- *Nature de l'environnement* Ce critère indique globalement les caractéristiques de l'environnement sur lequel les agents agissent. C'est dans cet environnement qu'un agent perçoit localement, et son comportement dépend de ce qu'il a perçu (y compris les autres agents). Ces caractéristiques vont permettre à un agent cognitif d'avoir une représentation explicite de l'environnement et des autres agents. Ainsi, il pourra régulariser et optimiser ses comportements en fonction des prédictions faites sur les comportements des autres agents. C'est pour cette raison que nous avons ajouté la valeur "explicite". Nous avons retenu les valeurs "structuré" et stable" de [ADA99a], et, "déterministe" et "observable" de [LOG98]. Cependant, un agent ne peut posséder toutes les connaissances sur l'environnement (la société dans laquelle il vit). La méthodologie tient-elle compte de ces caractéristiques pour concevoir l'architecture d'un agent?

- *structuré* : *indique si les rôles, les fonctions sont clairement définis.*
- *stable* : pouvoir faire face à des perturbations. Maintenir la valeur de certains paramètres constants; capacité de réorganisation.
- *déterministe :* un environnement pour lequel à partir de son état actuel, et de l'ensemble des actions qui peuvent y être exécutées, ses futurs états peuvent être prédits par un observateur averti. Dans le cas contraire, l'environnement est dit non déterministe [LOG98].
- explicite : qui est réellement exprimé ou formulé.
- *observable* : un environnement pour lequel il est possible de déterminer à chaque instant tous les états possibles, sinon il est partiellement observable [LOG98].

3- *Type de l'environnement* Actif, passif (base de données par exemple).

4- *Caractéristiques des données traitées dans l'organisation* Ce critère indique si la méthode peut considérer les données de façon :

- *qualitative (symbolique).*
- *quantitative (numérique).*

Une méthodologie qui permet de manipuler aussi bien les données numériques que symboliques est polyvalente et susceptible d'être applicable à un plus grand nombre de situations.

6.2.5 Dimension Coopération

La coopération est l'une des plus importantes rubriques des SMA où les agents doivent coopérer pour atteindre un objectif commun.

Plusieurs auteurs (par exemple [GOL94], [SEK95]) ont étudié l'influence du comportement social des agents sur la performance globale du système. Ils ont montré que la coopération entre agents améliore les résultats.

Hogg et Huberman [HOG92] ont montré que lorsque des agents coopèrent dans le cadre d'une résolution distribuée de problème, ils le résolvent plus rapidement que n'importe quel agent travaillant isolément.

Glize, Gleizes et Camps [GLI98] ont démontré que tout système coopératif est fonctionnellement adéquat. «Un système est fonctionnellement adéquat si son activité est «correcte» dans l'environnement dans lequel il est immergé. Il s'intègre, ainsi, durablement dans le monde ou en façonne un nouveau. L'activité correcte n'est décidable que par un observateur extérieur jugeant les interactions et connaissant la fonction que le système doit réaliser dans son environnement» [GLI98].

Il nous semble souhaitable de retrouver dans une méthodologie, des principes de coopération génériques, applicables de façon invariable pour tous les systèmes dont elle peut supporter la conception, sans qu'ils (les principes) ne dépendent de la fonction à obtenir. Par exemple, ces principes devraient permettre d'établir et de maintenir l'état coopératif au niveau des agents contenus dans le système. En d'autres termes, ces principes devraient permettre de résoudre des états non coopératifs tels que : l'incompréhension, l'ambiguïté, la concurrence, le conflit, etc.

Cette dimension permettra de voir l'existence de ces principes dans la méthodologie étudiée. Elle décrit les modèles de coopération, les modes de communication, les langages de communication et la nature des interactions utilisées dans la méthodologie.

Six critères ont été associés à cette dimension. Le critère "Communication" de [ADA99a] a été retenu ici et est intitulé "Mode de communication" pour être plus significatif. Les cinq autres critères sont tirés de [AFI98]. Les critères de la dimension Coopération sont :

1-*Types de communication* il s'agit de :

- *communication entre agents hétérogènes* : utilisation de modèles et de langages d'implémentation différents.
- *communication agents-humains.*

2- *Mode de communication* Ce critère indique la prise en compte de la communication de données, du mode de communication qui peut être :

- *direct* : *envoi de message*
- indirect : utilisation de tableau noir. Les agents accèdent à une base de données partagée appelée tableau noir dans laquelle les informations sont portées. Dans le système MINDS [HUH87], chaque usager travaille avec un agent qui utilise un tableau pour retrouver des documents dans sa base de données locale ou communique par message avec d'autres agents du même type.
- *synchrone* : *par téléphone par exemple.*
- *asynchrone* : *la messagerie par exemple.*

3- *Langage de communication* Ce critère indique si le langage utilisé par les agents peut être basé sur :

- *les signes* : par exemple le langage de communication des fourmis est basé sur les signes.
- *les actes de discours* : KQML par exemple.
- *autres* : il se peut qu'il existe un langage de communication qui soit différent des deux premiers.

4- Modèle de coopération Ce critère indique les concepts de coopération utilisés dans les modèles d'interaction de la méthode :

- négociation : parvenir à un accord acceptable pour tous les agents concernés. C'est un concept qui permet de résoudre les conflits entre agents.
- *délégation de tâches (facilitateurs)* : ce concept a été utilisé dans le MID où la gestion et le suivi des protocoles d'interaction sont complètement délégués au mécanisme d'interaction. Les agents n'auront plus à s'occuper de cette gestion (voir section 4.5).
- planification multi-agents (élaboration et exécution des plans possibles). Le Global Partial Planning (GPP) ([DUR87] [DUR89]) est une approche flexible qui permet aux divers agents d'un système de se coordonner dynamiquement [CHA99].

5- *Type de contrôle* Ce critère indique le type de coordination utilisé dans les modèles d'interaction de la méthode.

- *centralisé* : le contrôle est assuré par un seul agent (exemple : superviseur-employés).
- *hiérarchique* : c'est une direction dans laquelle il y a une série ascendante de pouvoirs ou de décisions (la coordination dans les entreprises par exemple).
- *distribué* : c'est une direction répartie; le processus de décision est conjoint (le marché par exemple).

Dans tous les cas de figure, les agents doivent être capables d'échanger entre eux des résultats intermédiaires et de faire des transferts de ressources. La coordination est une question centrale pour les SMA. Sans la coordination, un groupe d'agents peut dégénérer rapidement en une collection chaotique d'individus [CHA99].

6-Interaction *Ce critère indique si l'interaction est statique (exemple : message) ou dynamique [exemple : message actif (voir section 4.5)]. Il indique si le moteur d'interaction est distribué (interne à l'agent) ou centralisé (serveur ou médiateur). Ce critère indique aussi la nature des protocoles d'interaction utilisés dans la méthode, qui peut être explicite ou implicite.* À travers ce critère, on pourrait observer si le mécanisme d'interaction utilisé dans la méthode permet de résoudre les états non coopératifs entre agents.

6.2.6 Dimension Technologie

Cette dimension a pour but de décrire les caractéristiques des logiciels visés par la méthode. Ces caractéristiques constituent, à notre avis, un paramètre dans le choix d'une méthode convenable pour une application donnée. Un quatrième critère, type d'application visé,

a été ajouté aux trois critères retenus dans [ADA99a], pour cette dimension. Ce critère a été tiré des caractéristiques générales des plateformes SMA [AFI98]. Les critères sont :

1- *Mode de traitement* *Ce sont les modes de traitements (ou de fonctionnement) possibles des logiciels visés :*

- *batch (en lots)*
- *interactif*
- *client-serveur*
- *synchrone*
- *asynchrone*
- *distribué (modules distribués)*

2- *Type d'interface homme-machine* *Il existe différents types possibles d'interface :*

- *classique : Lorsque la méthode ne prend véritablement en considération l'interface utilisateur.*
- *adaptable ou flexible : dynamique; configurable par l'utilisateur (comme en VB ou en VC++).*
- *adaptative : l'interface vise à s'adapter automatiquement aux besoins de l'utilisateur.*
- *assistante : l'interface va raisonner en parallèle à l'utilisateur et se comporter comme un assistant humain.*

3- *Programmation* *Ce critère indique si elle doit être structurée, orientée-objet, orientée-agent.*

4- *Type d'applications visées*

- *simulation* : pour l'aide à la décision par exemple.
- *résolution de problème*
- *intégration*
- etc.

5- Environnement de développement Ce critère décrit les caractéristiques de développement des SMA auxquelles la méthode peut s'appliquer (plateformes possibles, langages de programmation, autres outils servant à implémenter les agents, etc.).

L'implémentation des agents est-elle déclarative ou procédurale? Quels sont les protocoles de communication utilisés dans le SMA (KQML, Http, HTML, OLE, etc.).

6.3 Évaluation des méthodologies

L'évaluation des méthodologies se fera autour de chaque dimension de ce cadre CaMuCCoSMA. Nous effectuerons cette évaluation à travers des grilles. La grille d'évaluation des méthodologies autour d'une dimension donnée est une matrice (tableau) $(A_{ij})_{ij}$ de format n×m (Figure 6.1) où l'indice de ligne i réfère aux valeurs des critères et l'indice de colonne j réfère aux méthodologies étudiées. n-2 est le cardinal de l'ensemble des valeurs de tous les critères de la dimension et m-2 est le nombre de méthodologies étudiées. La première colonne du tableau contient les critères de la dimension. La deuxième contient les valeurs de ces critères. Chacune des autres colonnes contient les résultats d'évaluation d'une méthodologie. La case d'un critère est divisée en p lignes, p étant le nombre de valeurs que ce critère possède. A_{ij} est l'élément qui apparaît à l'intersection de la ligne i et de la colonne j.

$A_{ij} \in$ {oui, non, possible, blanc} avec $i \neq 1$ et $i \neq 2$, et $j \neq 1$ et $j \neq 2$.

A_{ij} = oui indique que la méthodologie qui se situe en haut de la colonne j a pris en compte la valeur du critère qui se situe au bout et à gauche de la ligne i.

A_{ij} = non indique que la méthodologie ne prend pas en compte cette valeur.

A_{ij} = possible signifie qu'au regard de certains éléments présents dans la description de la méthodologie de la colonne j, on pourrait déduire qu'il y a une prise en compte implicite par cette méthodologie de la valeur de la ligne i.

A_{ij} = blanc signifie rien dans la documentation lue présentant la méthodologie de la colonne j ne nous permet d'affirmer que cette dernière possède la valeur du critère, de la ligne i. Le formulaire des grilles est celui utilisé dans [ADA99a]. Nous avons expliqué ici comment il faut faire la lecture de ces grilles.

Remarque

La grille d'évaluation des méthodologies autour de la dimension Représentation s'interprétera de la façon suivante :

- au niveau du critère "Nombre de modèles", A_{ij} est plutôt un entier quelconque et représente le nombre de modèles utilisés dans la méthodologie de la colonne j.
- au niveau du critère "Formalisme", A_{ij} indique plutôt les modèles de la méthodologie de la colonne j.
- au niveau du critère "Séquencement", A_{ij} indique plutôt comment les différents modèles de la méthodologie de la colonne j sont interreliés; $M_1 \rightarrow M_2$ signifie que le modèle M_2 dérive du modèle M_1. La définition des modèles et la valeur de A_{ij} (représentée dans la grille par le symbole &j) se trouveront dans le commentaire de la méthodologie qui suivra la grille d'évaluation. Le nombre qui apparaît dans la ligne Complexité des modèles est le nombre d'interrelations entre les modèles.

		Méthodologies																	
Critères	**Valeurs des critères**	**M1**	**M2**	**....**	**....**	**Mj**													
	V1																		
	V2																		
	Vi					A_{ij}													

Tableau 6.1 : Grille d'évaluation des méthodologies autour d'une dimension.

6.4 Conclusion

Ce chapitre décrit le cadre CaMuCCoSMA qui nous a servi de repère pour faire l'étude comparative des méthodologies SMA présentées dans le chapitre 7. Ce cadre contient six dimensions. Chaque dimension est définie par un certain nombre de critères. Les critères possèdent des valeurs qui permettent d'analyser les méthodologies. L'étude comparative a été réalisée autour de chaque dimension à travers une grille. Afin de rendre simple l'exploitation de CaMuCCoSMA, nous l'avons décrit dans la syntaxe du langage CML (Conceptual Modeling Language) [SCH94a].

Quatrième partie

Étude comparative des méthodologies SMA

Chapitre 7

Étude Comparative des Méthodologies SMA selon CaMuCCoSMA

Dans ce chapitre, nous avons appliqué le cadre CaMuCCoSMA à neuf méthodologies. Il s'agit de :

1. La méthodologie GAIA [WOO00].
2. Multiagent Systems Engineering (MaSE) [SCO99].
3. A Methodology and Modeling Technique for Systems of BDI Agents (MMTS) [KIN96].
4. An Agent-Oriented Methodology : High-Level and Intermediate Models (HLIM) [EAL99].
5. The CoMoMAS (Conceptual Modeling of Multi-Agent systems) Methodology and Environement for Multi-Agent System Development [NOR96].
6. Multi-Agent Scenario-Based Method (MASB) [BER96].
7. Analysis and Design of Multiagent Systems Using MAS-CommonKADS [CAR98].
8. Agent-Oriented Methodology for Enterprise Modeling (AOMEM) [KEN96].
9. Agent-Oriented Design of Soccer Robot Team (Cassiopée) [ANE96].

Nous avons analysé ces neuf méthodologies suivant chaque dimension du cadre CaMuCCoSMA. Cette analyse est suivie de discussions partielles et d'une discussion générale faisant ressortir les points communs, les avantages et les insuffisances de ces méthodologies.

7.1 Comparaison des méthodologies selon CaMuCCoSMA

7.1.1 Dimension méthodologie

Le tableau 7.1 donne une vue sur les étapes et l'approche utilisées dans la méthodologie étudiée. Il indique également la prise en compte de l'utilisateur dans la méthode.

Légende

Oui : signifie que la méthodologie a pris en compte cette valeur.

Non : signifie que la méthodologie n'a pas pris en compte cette valeur.

Blanc : signifie qu'on ne peut rien conclure sur la base des documents que nous avons lus.

Possible : signifie qu'au regard des éléments décelés dans les documents, on peut déduire que la méthodologie pourrait prendre en compte cette valeur.

Les valeurs marquées dans les cases au niveau de support logiciel représentent les outils utilisés.

Critères	Valeurs des critères	MÉTHODOLOGIES								
		GAIA	MaSE	MMTS	HLIM	CoMoMAS	MASB	MAS-CommonKADS	AOMEM	Cassiopée
Étapes du processus	analyse	oui	oui	oui	oui	oui	oui	oui	oui	oui
	modélisation	oui	oui	oui	oui	oui	oui	oui	oui	oui
	spécification	oui	oui	oui	oui	oui	oui	oui	oui	oui
	conception	oui	oui	oui	oui	oui	oui	oui	oui	oui
	validation	non	non	non	non	possible	oui			possible
	vérification	non	possible	oui	possible	oui	non	oui		oui
	évaluation ergonomique	non	possible	possible	possible	possible	possible	oui	possible	non
Modèles de développement	cascade	oui	non	non	non	non	non	non	non	non
	incrémentale	non	oui	oui	non	oui	non	non		non
	spirale	non		non	oui	non	oui	oui		oui
	V	non	non	non	non	non	non	non		non
	nabla	non	non	non	non	non	non	non		non
Approche de développement	descendante	oui	oui	oui	oui	oui	non	oui	oui	oui
	ascendante	oui	oui	oui	non	oui	non	oui		oui
	évolutive	non	oui	non	non	possible	oui	non	non	non
Degré d'implication de l'utilisateur	faible	oui	oui	oui	oui	oui	non	oui	oui	oui
	moyen	non	non	non	non	non	non	non	non	non
	forte	non	non	non	non	non	oui	non	non	non
Moment d'implication de l'utilisateur	début	oui	oui	oui	oui	oui	oui	oui	oui	oui
	milieu	non	non	non	non	non	oui	non	non	non
	fin	oui	oui	oui	oui	oui	oui	oui	oui	oui
Réutilisabilité		oui	oui	possible	oui	oui	oui	possible		possible
Disponibilité de support logiciel	Analyse	FUSION	AgML	OMT	UCMs	CML, CommonK ADS	SMAU L2	OOSE, MSC, RDD, SDL, HMSC, CML, OMT, CommonKADS	IDEF, OOSE	OMT
	modélisation	FUSION	AgML	OMT	UCMs	CML, CommonK ADS	SMAU L2	OOSE, MSC, RDD, SDL, HMSC, CML, OMT, CommonKADS	IDEF, OOSE	OMT
	spécification	FUSION	AgML	OMT	UCMs	CML, CommonK ADS	SMAU L2	OOSE, MSC, RDD, SDL, HMSC, CML, OMT, CommonKADS	IDEF, OOSE	OMT
	conception	FUSION	AgML,AgDL, UML	OMT	UCMs	MICE	SMAU L2		IDEF, OOSE, Ontolingua	OMT
	validation									
	vérification		AgDL							
	évaluation ergonomique									

Tableau 7.1 : Aspects méthodologiques préconisés par les méthodes

A) Commentaires

GAIA

GAIA couvre presque totalement toutes les étapes du processus de développement du projet à l'exception des phases de vérification, de validation et d'évaluation ergonomique. La méthodologie GAIA supporte le développement en cascade et possède une approche mixte. L'utilisateur intervient au début et à la fin du projet. Les modèles de GAIA sont génériques et peuvent donc être réutilisés. Les propriétés de permission d'un rôle sont exprimées dans une notation formelle basée sur la notation de FUSION [COL94]. FUSION représente, donc, un support logiciel pour la méthodologie GAIA. GAIA utilise aussi les notations de la technologie objet.

MaSE

À part l'étape de la validation, MaSE couvre toutes les autres étapes du processus. La méthodologie MaSE utilise le modèle de développement incrémental et possède une approche évolutive. En effet, les quatre premières phases de la méthodologie, à savoir Conception du Domaine, Conception d'Agent, Conception de Composante et Conception du Système constituent une extension progressive des fonctionnalités du système. La première phase, Conception du Domaine, consiste à identifier les types (classes) d'agents, à définir leurs interactions possibles et les protocoles de coordination pour chaque type. Ensuite, on descend successivement aux niveaux de la phase de Conception d'Agents (définition de l'architecture de chaque type d'agent) et de la phase de Conception de Composantes (conception des composantes d'un agent). Puis au niveau de la dernière phase, Conception du Système, on remonte à la conception de l'architecture globale du système. L'utilisateur intervient au début et à la fin du processus. MaSE utilise les techniques des cas d'utilisation pour identifier les agents. Il est à noter que MaSE a été construite sur la base des techniques de conception orientée-objet telles que Object Modeling Technique (OMT) [RUM91] et Unified Modeling Language (UML) [MUL97]. La méthodologie MaSE est supportée par les outils Agent Modeling Language (AgML) et Agent Definition Language (AgDL) [SCO99]. AgML est un langage graphique qui permet de décrire les types d'agents d'un système et leurs interactions avec d'autres types d'agents. AgDL est un langage basé sur la logique des prédicats du premier ordre et est utilisé pour décrire complètement les comportements internes de chaque agent. Les sémantiques et les

syntaxes formelles de AgML et de AgDL permettent une réutilisation automatique des composantes et facilitent la vérification.

MMTS

À l'exception de la validation, la méthodologie MMTS couvre toutes les étapes du processus de développement. La prise en compte de l'évaluation ergonomique est possible car le modèle d'interaction décrit aussi l'interaction système-utilisateur. Elle utilise le modèle de développement incrémental et suit une approche mixte et itérative. En effet, MMTS définit deux principaux niveaux :

- Un niveau externe qui consiste à décomposer le système en agents qui sont modélisés comme des objets complexes caractérisés par leurs objectifs, leurs responsabilités, les services qu'ils exécutent, les informations qu'ils demandent et maintiennent, et la définition de leurs interactions.

- Un niveau interne qui fait ressortir la modélisation de chaque classe agent BDI à travers trois modèles qui sont : le modèle de croyances, le modèle d'objectifs et le modèle de plans.

L'utilisateur apparaît seulement au début dans la méthodologie MMTS. Ses modèles devraient, en principe, être réutilisables. Dans l'architecture BDI de MMTS, un agent peut être complètement spécifié par les événements qu'il peut percevoir, les actions qu'il peut exécuter, les croyances qu'il peut tenir, les objectifs qu'il peut adopter et les plans qui peuvent apparaître dans ses intentions. MMTS utilise la notation OMT.

HLIM

La méthodologie HLIM couvre toutes les étapes du processus de développement sauf l'étape de la validation, car l'utilisateur apparaît seulement au début du projet pour présenter ses exigences. HLIM supporte le modèle spiral et possède une approche descendante. HLIM possède deux phases :

1. La phase de découverte qui permet d'identifier les agents du système et leurs niveaux de comportement. Elle a pour but de produire un modèle appelé Hight-Level Model (HLM) qui capture la structure et le comportement du système.

2. La phase de définition qui produit des définitions implémentables. Elle a pour but d'avoir une compréhension claire des comportements des entités qui participent à leur exhibition

et des rapports entre ces entités. Cette phase se réalise à travers 4 modèles. Les cinq modèles de HLIM sont génériques et sont applicables à différents systèmes. La méthodologie HLIM est supportée par les UCMs (Uses Case Maps) [DAN99] et par la représentation textuelle des UCMs qui est basée sur le langage Extented Markup Langage (XML). Le navigateur UCMNav permet de générer les UCMs.

CoMoMAS

La méthodologie CoMoMAS couvre presque toutes les étapes du processus de développement. L'étude ergonomique est possible à cause de l'utilisation dans CoMoMAS de CLOS [BER93] qui est un langage de programmation orientée-objet à interface utilisateur graphique. CoMoMAS est supporté par le modèle incrémental et possède une approche mixte. Rien ne nous indique l'implication complète de l'utilisateur dans la méthodologie.

CoMoMAS est conçu avec la réutilisabilité et pour la réutilisabilité, c'est- à dire que ses modèles sont réutilisables et sont conçus en utilisant des bibliothèques existantes. En effet, elle constitue une extension de la méthodologie à base de connaissances CommoDADS [SCH94b] aux SMA. CommonKADS fournit une librairie structurée et flexible de modèles qui supportent CoMoMAS. Les modèles de CoMoMAS sont représentés dans la version étendue du langage CML (Conceptual Modeling Language) [SCH94a]. CML est un langage semi-formel qui permet de décrire les connaissances des agents. La validation et la vérification sont supportées par la version étendue de MICE (Michigan Intelligent Coordination Experiment [MON90]). MICE est un environnement qui fournit des mécanismes d'évaluation pour la validation expérimentale des architectures d'agents et des systèmes en simulation.

MASB

À part l'étape de vérification, MASB couvre toutes les autres étapes du processus de développement d'un logiciel. MASB est supportée par le modèle spiral et possède une approche évolutive. L'utilisateur est moyennement impliqué dans le processus du développement (il intervient seulement pendant les phases d'analyse, de spécification et de conception). La générécité des modules de MASB permet la réutilisation de ces derniers. Ces modules sont générés par l'environnement SMAUL2 pour Systèmes Mutli Agents Université Laval [BER96].

MAS-CommonKADS

La méthodologie MAS-CommonKADS couvre toutes les étapes du processus de développement à part l'étape de la validation. L'évaluation ergonomique se remarque à travers le modèle de communication. La prise en compte de l'interaction homme-machine a été l'une des extensions faites à la méthode CommonKADS. MAS-CommonKADS est supportée par le modèle spiral et possède une approche mixte. Le système est partiellement testé en détectant, par les méthodes de résolution de conflits, les conflits entre les scénarios. Nous ne pouvons affirmer l'implication de l'utilisateur tout au long du processus. Comme CoMoMAS, MAS-CommonKADS est conçu en utilisant des bibliothèques existantes. Ses modèles sont réutilisables. Elle utilise les techniques de la méthode à base de connaissances CommonKADS [SCH94], les techniques des méthodologies orientées-objet telles que Object Modeling Technique (OMT) [RUM91], Object Oriented Software Engineering (OOSE) [JAC92] et Responsibility Driven Design (RDD) [WIL90]. Elle utilise aussi les langages Specification and Description Language (SDL) [ITU94], Message Sequence Charts (MSC96) [EKK96], High level Message Sequence Charts (HMSC) [ITU94] et CML [SCH94a] pour spécifier les protocoles d'interactions des agents. SDL et MSC permettent le test. Les modèles de MAS-CommonKADS sont génériques c'est- à dire instanciables.

AOMEM

La méthodologie AOMEM supporte les étapes d'analyse, de modélisation, de spécification et de conception du processus de développement. Cependant ces étapes ne sont pas clairement séparées. On ne peut rien conclure sur les modèles qui la supportent. Le processus de développement n'est pas clair. Elle possède une approche descendante. L'implication de l'utilisateur n'est pas traitée. Rien ne nous permet, dans la documentation lue, de dire si les modèles sont réutilisables. Cette méthodologie combine les méthodologies orientées-objet telle que Object Oriented Software Engineering (OOSE) [JAC92] et les méthodologies de modélisation d'entreprise telle que Integration Definition for Enterprise Function (IDEF) [FIP93]. Tout porte à croire que les langages de communication utilisés par les agents sont COOL, qui est une extension de KQML [BAR95], et AgenTalk [KUW95].

Cassiopée

À l'exception de l'évaluation ergonomique, Cassiopée couvre toutes les étapes du processus de développement. Cependant, il est difficile d'identifier clairement ces étapes. Cette méthodologie est supportée par le modèle spiral et possède une approche mixte. L'implication de l'utilisateur n'est pas indiquée dans la documentation décrivant cette méthodologie. La réutilisation des modèles fournis par Cassiopée est possible car ses auteurs pensent l'intégrer aux méthodes d'analyse existantes.

Cette méthodologie se place dans le contexte du projet de recherche MICROB (Making Intelligent Collective Robotics), dont l'objectif est d'investiguer les phénomènes d'organisation collective dans des sociétés de robots. Pour la formation dynamique de groupes, Cassiopée utilise les techniques existantes [DEC87] telles que les techniques dérivant des réseaux de contrats [SMT80], les techniques basées sur la notion de consensus et de négociation entre les agents appartenant à des groupes concurrents [SYC89], etc.

B) Discussion

À part les étapes de validation, de vérification et d'évaluation ergonomique, toutes les méthodes supportent les autres étapes du processus même si ces dernières ne sont pas explicitement séparées dans les méthodes MMTS, CoMoMAS, AOMEN et Cassiopée. MAS-CommonKADS prend en compte la vérification (même si celle-ci n'est pas très explicite) et l'évaluation ergonomique. Le modèle de développement utilisé et l'approche suivie n'ont pas été clairement indiqués dans les méthodes étudiées. À part MASB, l'utilisateur intervient seulement au début et à la fin du processus de développement des systèmes que les méthodologies peuvent concevoir. Toutes les méthodes fournissent, ou du moins ont un souci de fournir, des modèles réutilisables et possèdent toutes des supports logiciels ou méthodologiques. En particulier, MAS-CommonKADS possède des supports logiciels et méthodologiques très variés, ce qui à notre avis renforce un peu plus cette méthodologie.

7.1.2 Dimension représentation

Le tableau 7.2 indique globalement le formalisme utilisé pour représenter et modéliser le système à étudier.

Critères	Valeurs des critères	MÉTHODOLOGIES GAIA	MaSE	MMTS	HLIM	CoMoMAS	MASB	MAS-Common KADS	AOMEM	Cassiopée
Découpage du système	niveaux d'abstraction	oui	oui	oui	oui	oui	oui	oui	oui	oui
	généralisation/ spécialisation	oui	oui	oui	oui	oui	oui	oui	oui	oui
	type/occurrence	oui	oui	oui	oui	oui	oui	oui	oui	oui
	stratégie/tactique	non	non	non	non	non	non	oui	possible	non
Formalisme	données	MR, MI, MA	DCC, DA	MA, MC, MO	HLM, IAM	MCo, MT, ME, MA	DS, MCD, DSDE, MO, MA	ME, MA, MO, MCo	MCU, MD, MF, SOA	EB,RB,OB
	traitements	MI, MS	DCC	MI, MP	HLM, IAM, CvM, ARM, CtM	MC, ME, MS, MA, MCo	DFR MA	MA, MT, ME	MF, SOA	EB,RB,OB
	activités	MR, MS	DA	MI,MC	HLM, IAM	MT, MS	MA, DFR	MA, MT, ME	MF, SOA	EB,RB,OB
	dynamique	MI, MRe	DA, DCC, DHC	MI,MP	HLM, CvM, ARM, CtM	MS, MCo	MIUS, MC, MA	MCo, MCn, MC	MD, SOA	OB
Séquencement		&1	&2	&3	&4	&5	&6	&7	&8	&9
Qualité des modèles	nombre de modèles	5	4	5	5	6	9	7	4	3
	cohérence des modèles	possible	possible	possible	possible	possible	possible	possible	possible	possible
	complétude des modèles	non	non	non	non	non	non	possible	non	
	complexité des modèles	6	6	10	6	8	13	15	6	3

Tableau 7.2 : Vue d'ensemble du système que la méthodologie étudiée peut concevoir

Légende

Oui : signifie que la méthodologie a pris en compte cette valeur.

Non : signifie que la méthodologie n'a pas pris en compte cette valeur.

Blanc : signifie qu'on ne peut rien conclure sur la base des documents que nous avons lus.

Possible : signifie qu'au regard des éléments décelés dans les documents, on peut déduire que la méthodologie pourrait prendre en compte cette valeur.

Le symbole &j dans la ligne Séquencement indique les liens entre les modèles de la méthodologie concernée (sa valeur est indiquée en dessous du tableau, au niveau des commentaires). $M_1 \rightarrow M_2$ signifie que M_2 dérive de M_1. Le nombre qui apparaît dans la ligne Complexité des modèles est le nombre d'interrelations entre les modèles.

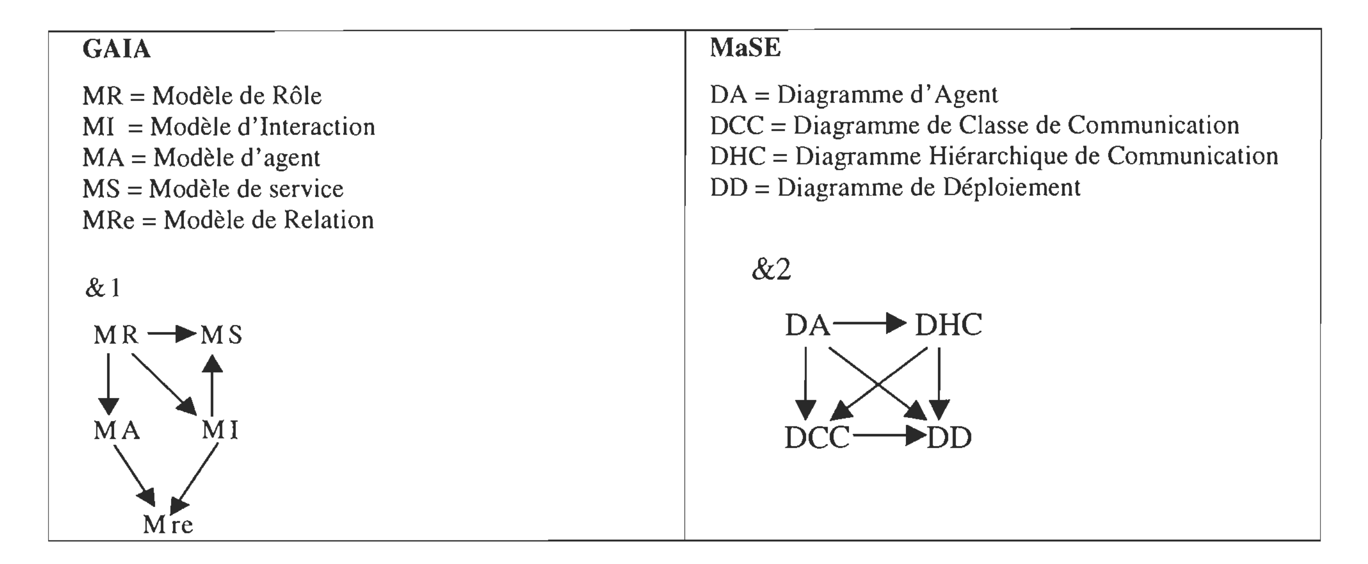


Figure 7.1 : Séquencement de GAIA **Figure 7.2 : Séquencement de MaSE**

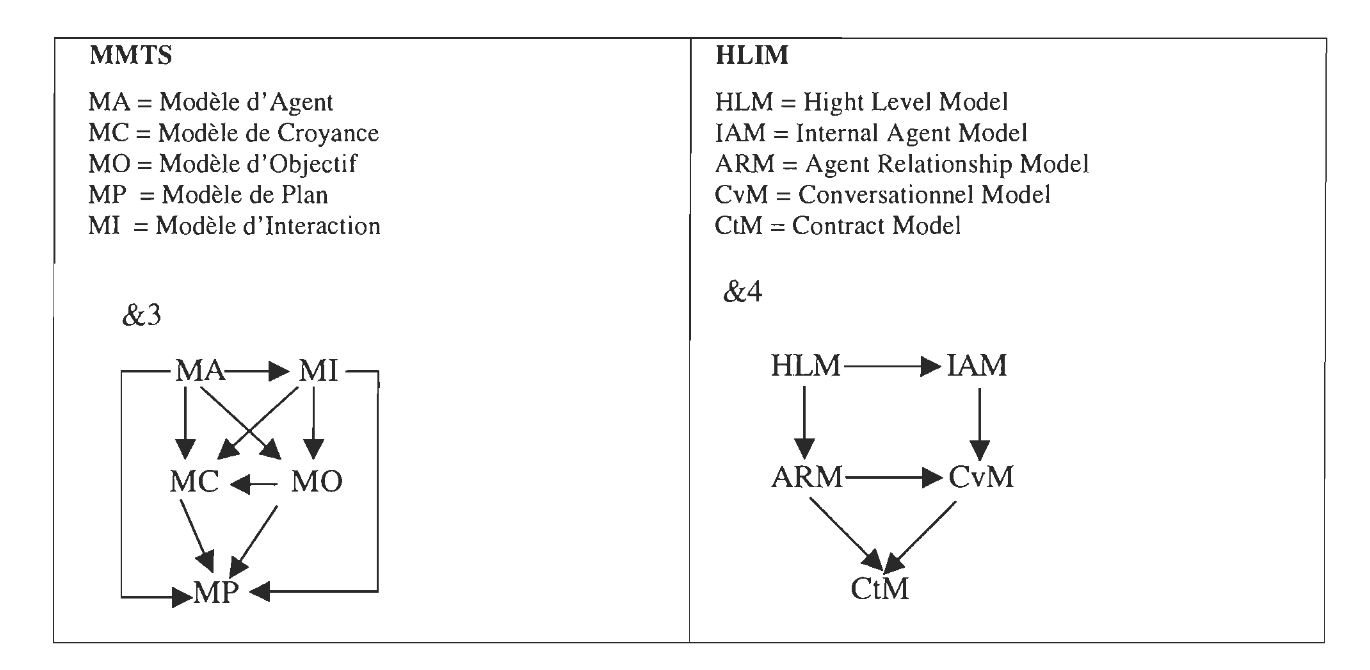


Figure 7.3 : Séquencement des modèles MMTS **Figure 7.4 : Séquencement des modèles de HLIM**

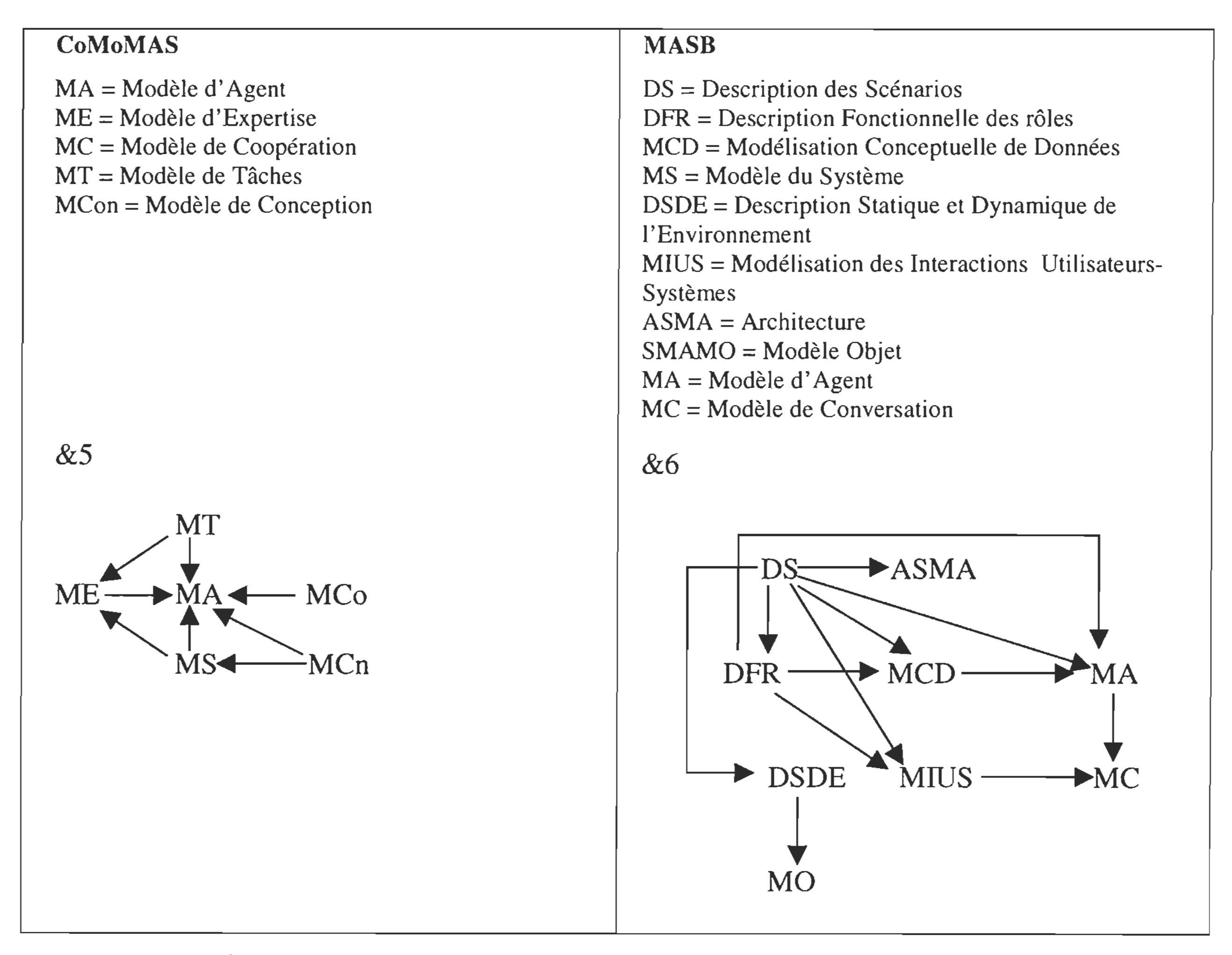


Figure 7.5 : Séquencement des modèles de CoMoMAS

Figure 7.6 : Séquencement des modèles de MASB

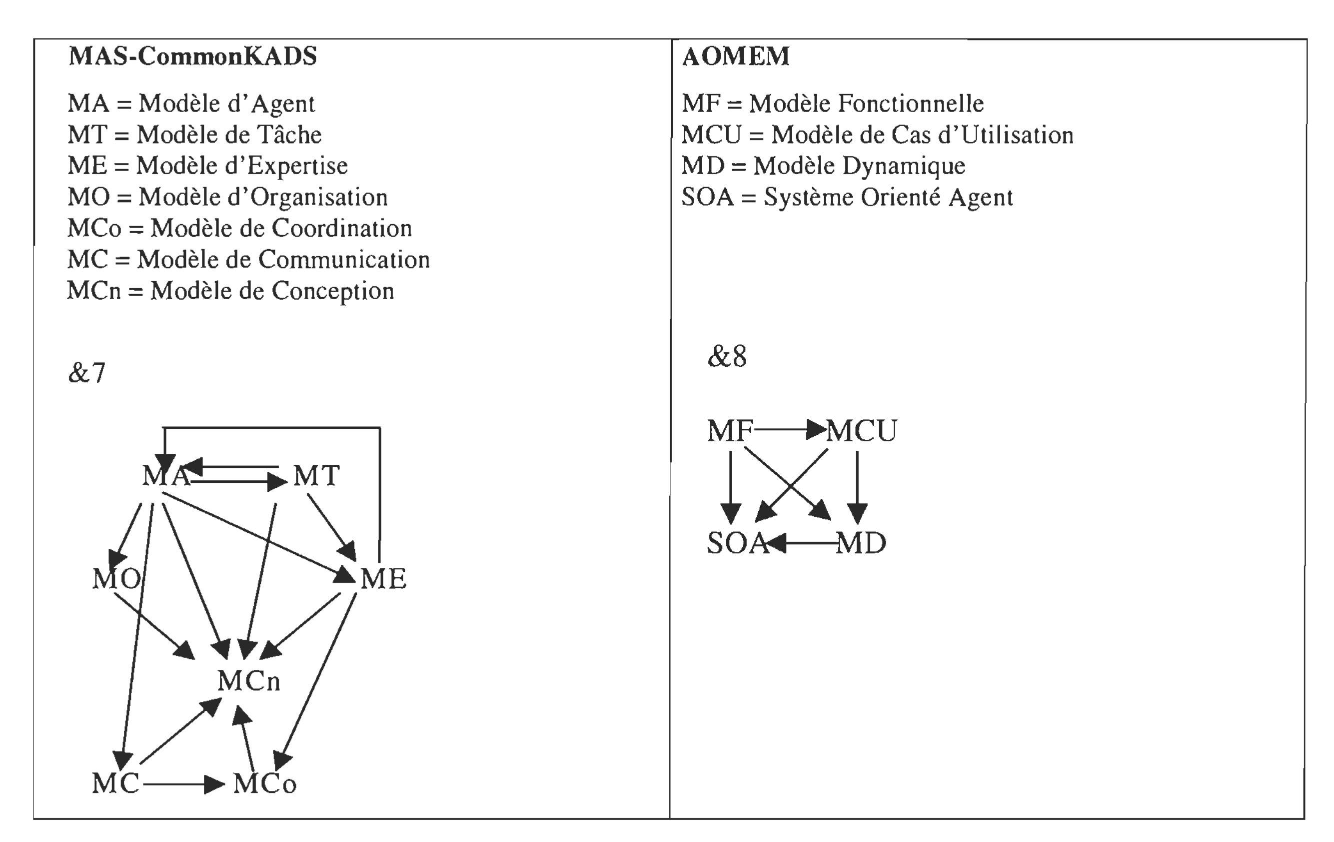


Figure 7.7 : Séquencement des modèles de MAS-CommoKADS

Figure 7.8 : Séquencement des modèles de AOMEM

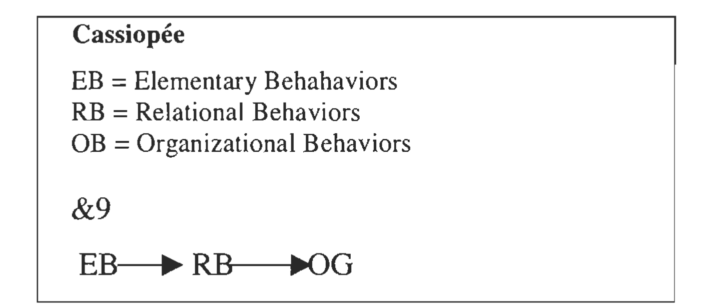


Figure 7.9 : Séquencement des modèles de Cassiopée

A) Commentaires

GAIA

La méthodologie GAIA contient 5 modèles qui sont :

- le Modèle de Rôle (MR) : identifie les principaux rôles du systèmes;

- le Modèle d'Agent (MA) : identifie les types d'agents du système et les agents qui vont instancier ces types;
- le Modèle de Service (MS) : identifie les principaux services nécessaires pour jouer les rôles des agents. Un service étant un bloc cohérent d'activités dans lequel un agent s'engagera.
- le Modèle Relationnel (MRe) : documente les lignes de communications entre les différents agents;
- le Modèle d'Interaction : qui est un ensemble de définitions de protocoles d'interaction.

Le découpage en niveaux d'abstraction est marqué par les concepts de rôle et d'agents utilisés dans la méthodologie. La notion de généralisation/spécialisation est marquée par le fait que certains rôles peuvent être paramétrables. Le modèle d'agent identifie les types d'agents du système et les agents qui vont instancier ces types.

La méthodologie GAIA traite les données (les agents, les objets par exemples) à travers les modèles de rôle et d'agent; les activités (unité de tâche) à travers le modèle de rôle; les traitements (un traitement étant vu comme une série d'opérations logiques pour exécuter une tâche donnée) à travers les modèles de services et d'interaction; la dynamique à travers les modèles d'interaction et de relation. Le modèle d'interaction, constitué des protocoles d'interaction, peut être dérivé des exigences de l'utilisateur et du modèle de rôle. Mais le modèle d'interaction est utilisé pour raffiner le modèle de rôle. Le modèle de rôle dérive du domaine d'application. Le modèle d'agent est dérivé du modèle de rôle. Le modèle de service peut être dérivé des modèles d'interaction et de rôle. Le modèle de relation (qui est un graphe dirigé dans lequel les nœuds constituent les types d'agent et les arcs, les chemins de communication) peut être dérivé des modèles d'interaction et d'agent. Il nous semble qu'il y a peu de redondance dans le séquencement de ces modèles qui sont, à notre avis, incomplets car les modèles de rôle et d'interaction ne prennent véritablement pas en compte les structures et les règles organisationnelles. Un modèle de rôle décrit tous les rôles d'une organisation et leurs positions dans cette dernière. Le modèle de rôle devrait être dérivé d'une structure organisationnelle qui est explicitement choisie par le concepteur. Les structures organisationnelles devraient être vues comme une première classe d'abstraction dans la conception d'un SMA. La définition d'une

structure organisationnelle d'un système peut être dérivée, entre autres des spécifications pendant la phase d'analyse [ZAM00]. La phase d'analyse devrait permettre d'identifier comment une organisation doit fonctionner. La phase de conception devrait définir quel type d'organisation est convenable aux spécifications définies pendant la phase d'analyse. La phase de conception devrait aussi permettre d'identifier clairement les règles d'engagement, de fonctionnement et les contraintes dans une organisation. Par exemple, dans les organisations humaines, les conventions sociales définissent un ensemble de règles implicites qui permettent de gérer les interactions entre leurs membres. Les conventions spécifiques d'une compagnie pourraient imposer des contraintes sur comment les différents rôles devraient être assurés dans chacune de ces organisations [ZAM00].

MaSE

La méthodologie MaSE contient 4 diagrammes. Les Diagrammes d'Agent (DA) définissent les différents types d'agents du système et les chemins de communication possibles (entre ces types d'agents) qui sont définis comme des conversations entre agents du système. L'approche utilisée ici est une approche systémique, c'est-à-dire qu'un type d'agent est représenté par une classe ayant un nom, des services et des objectifs. Une conversation est une séquence de messages entre deux agents qui sont impliqués dans la conversation. Les Diagrammes Hiérarchiques de Classes (DHC) définissent les rapports entre les différentes classes de conversations qui s'opèrent dans le système. Les Diagrammes de Classes de Communication (DCC), qui sont un ensemble de diagrammes de machines d'états finis (MEF), définissent les états de conversations dans lesquels chaque agent peut se trouver pendant l'exécution du système. Ces états sont définis par le rôle de l'agent intervenant dans la conversation. Chaque côté de la conversation est défini par un MEF séparé, donc deux MEF sont nécessaires pour définir complètement un DCC. Les Diagrammes de Déploiement (DD) définissent un SMA basé sur les agents issus des trois premières phases. C'est exactement comme les diagrammes de déploiement de UML. Le découpage du système en niveaux d'abstraction utilise les concepts de rôle, d'agent, de conversation (qui constitue un lien d'association entre classes d'agents). À travers les DHC, on note l'utilisation du principe de généralisation/spécialisation. L'utilisation de la relation classe-instanciation est remarquée dans les DD.

La méthodologie MaSE traite les données à travers les DA et DCC, les traitements à travers les DCC, les activités à travers les DA, et la dynamique à travers les DA, DCC, DHC. Les sémantiques de AgML et AgDL sont basées sur une approche algébrique, et permettent donc de générer automatiquement le code à partir des spécifications algébriques. Ce qui permet la réutilisabilité automatique des composantes et rend facile la vérification.

DA dérive des besoins de l'utilisateur, DHC dérive de DA, DCC dérive de DHC et de DA, DD dérive de DA, DHC et de DCC. Il nous semble qu'il y a peu de redondance dans le séquencement de ces modèles qui sont, à notre avis, incomplets car il manque des concepts de réorganisation de la société d'agents.

MMTS

La méthodologie MMTS contient 5 modèles. Un modèle d'agent (MA) identifie les classes d'agents (par l'identification des rôles), leurs instances qui peuvent exister à l'intérieur du système, et décrit les rapports hiérarchiques entre ces classes en utilisant les principes d'héritage et d'agrégation. L'approche systémique n'est pas appliquée ici car une classe est représentée seulement par son nom et ses attributs. Le formalisme utilisé pour représenter une classe d'agents est celui des objets dans la méthode Merise. Un modèle d'interaction (MI) décrit les responsabilités d'une classe d'agents, les services qu'ils fournissent, les interactions entre les agents, l'interface utilisateur et le contrôle des rapports entre les classes d'agents. À chaque classe d'agents, on peut associer les modèles de croyance, d'objectif et de plan par des attributs croyances, objectifs et plans. Un modèle de croyance (MC) décrit l'information sur l'environnement et l'état interne qu'un agent de cette classe peut avoir, et les actions qu'il peut exécuter. Ce modèle est constitué d'un ensemble de croyances et d'un ou de plusieurs états de croyance qui sont spécifiés par des diagrammes d'objets (qui définissent le domaine de croyance d'un agent) et l'algèbre des prédicats et des fonctions. Un modèle d'objectif (MO) décrit les objectifs qu'un agent peut adopter et les événements auxquels il peut répondre. Ces objectifs et événements sont définis dans un ensemble d'objectifs. Le formalisme utilisé pour définir ce modèle d'objectif est l'algèbre des prédicats et des fonctions. Un modèle de plan (MP) décrit les plans qu'un agent peut utiliser pour atteindre ses objectifs. Un ensemble de plans est défini pour décrire les propriétés et la structure de contrôle de chaque plan. Ce modèle est élaboré par une notation graphique similaire aux diagrammes de plans de Harel [HAR87] (ces diagrammes

ressemblent aux diagrammes d'états de l'orienté objet). Les concepts de rôle, de responsabilité, de service, et d'objectif sont les principales abstractions utilisées par MMTS pour maîtriser la complexité des systèmes. Le principe de généralisation/spécialisation est utilisé à travers le modèle d'agent. Celui de type/occurrence est utilisé aussi bien à travers le modèle d'agent qu'à travers les modèles de croyance et d'objectif. MMTS traite les données à travers les modèles MA, MC et MO; les traitements et la dynamique du système à travers les modèles MI et MP; et les activités à travers les modèles MI et MC. L'ordre de dérivation n'est pas explicite dans MMTS. Il nous semble cependant, d'après ce que nous avons lu, que : MA dérive du domaine d'application et des rôles du système; MI dérive de MA et est utilisé pour raffiner MA; MO dérive de MA et de MI; MC dérive de MA, MI, MO; et enfin MP dérive des 4 modèles MA, MI, MO, MC. Il nous semble qu'il y a un peu de redondance dans le séquencement de ces modèles qui sont, à notre avis, complexes (le nombre d'interrelation des modèles étant 10) et incomplets car les structures organisationnelles et les règles organisationnelles sont implicites.

HLIM

La méthodologie HLIM possède 5 modèles au total :

High Level Model (HLM) identifie les agents du système ainsi que leurs comportements, décrit le comportement global du système au niveau de la collaboration des agents et donne un catalogue de "plug-ins" décrivant quand et où ils peuvent être utilisés.

Internal Agent Model (IAM) décrit dans une table la structure interne des agents découverts dans le modèle HLM. Les agents sont décrits en terme d'objectifs, de croyances, de plans et de tâches.

Agent Relationship Model (ARM) décrit les rapports inter-agents. Ces rapports peuvent être de dépendance et de juridiction et sont décrits respectivement par les diagrammes de dépendance et les diagrammes de juridiction. Un diagramme de dépendance relate un agent qui fournit un service à un autre qui en a besoin. Un diagramme de juridiction décrit l'organisation des agents en terme de leur statut d'autorité. Les agents sont placés dans une hiérarchie juridictionnelle avec un agent en tête.

Converational Model (CvM) identifie les messages qui sont échangés par les agents lors de leur coopération et de leur négociation.

Contract Model (CtM) définit les prévisions sur la manière dont les agents peuvent satisfaire les rapports de dépendance ainsi que les attentes des agents lorsqu'ils jouent les rôles définis par le diagramme de juridiction.

Les principaux concepts d'abstraction utilisés dans cette méthodologie sont : les concepts de rôle, de responsabilité et d'objectif. Les principes de généralisation/spécialisation et de type/occurrence sont respectivement remarqués dans l'utilisation des «stack» et «stub» en pointillés et des «stack» et «stub» pleins des UCMs [DAN99]. Les UCMs constituent le formalisme utilisé pour la représentation et la modélisation des résultats issus de l'analyse.

HILM traite les données et les activités à travers les modèles HLM et IAM, les traitements à travers tous les 5 modèles et la dynamique du système à travers tous les modèles sauf IAM.

Dans la méthodologie HLIM, le séquencement des modèles est clairement indiqué. Ainsi HLM dérive du domaine d'application, IAM et ARM dérivent de HLM, CvM dérive de ARM, CtM dérive de ARM et de CvM. Le raffinement des 4 modèles de la phase de définition donne naissance à ce que les auteurs de HLIM ont appelé Agent Logique. Il nous semble qu'il y a peu de redondance dans le séquencement de ces modèles qui sont, à notre avis, incomplets car il manque des structures organisationnelles définies dans [ZAM00].

CoMoMAS

CoMoMAS est composé des 6 modèles suivants :

- le Modèle d'Agent (MA) décrit l'architecture et la structure des connaissances des agents (connaissance sociale, coopérative, de contrôle, cognitive et réactive). C'est le principal modèle;
- le Modèle d'Expertise (ME) décrit les compétences cognitives et réactives des agents;
- le Modèle de Coopération (MC) décrit la coopération entre agents, en utilisant les méthodes de résolution des conflits;
- le Modèle de Tâche (MT) décrit la décomposition des tâches du système et indique si ces tâches sont exécutées par un humain ou un agent informatique;

- le Modèle du Système (MS) définit les aspects organisationnels de la société des agents et leurs aspects architecturaux;
- le Modèle de Conception (MCo) intègre les modèles précédents dans un système global. Il décrit les exigences pour la conception.

Le découpage en niveaux d'abstraction dans la méthodologie est marqué par les concepts de rôle, de tâche et d'agent.

CoMoMAS traite les données à travers les modèles MCo, MT, ME, MS et MA; les traitements à travers les modèles MCo, MC, ME, MS et MA ; les activités à travers MT et MS et la dynamique du système à travers MS et MC.

Le séquencement des modèles est clairement indiqué dans CoMoMAS. Ainsi, MCo est dérivé de l'analyse des besoins, MT est dérivé de l'analyse des fonctionnalités du système, MS dérive de Mco, MC dérive de MS, ME dérive de MT et de MS, MA dérive des 5 autres modèles. Il nous semble qu'il y a peu de redondance dans le séquencement de ces modèles qui sont à notre avis un peu complexes (le nombre d'interrelations entre les modèles étant 8). Nous ne pouvons conclure sur la complétude des modèles.

MASB

La phase d'analyse de la méthodologie MASB est composée de :

- DS (Description des Scénarios) : identification (textuelle) des scénarios du point de vue de l'utilisateur, des principaux rôles des agents humains et informatiques et des objets de l'environnement;
- DFR (Description Fonctionnelle des Rôles) : description des rôles des agents utilisant les Diagrammes de Comportements (DC) qui décrivent le traitement, les informations pertinentes et les interactions entre agents. Un DC est associé à un rôle joué par un agent;
- MCD (Modélisation Conceptuelle des Données) : modélisation des données et des connaissances utilisées par les agents. Cette modélisation utilise les diagrammes rapport-entités (diagrammes orienté-objet) et les diagrammes de cycle de vie des entités;

- DSDE (Description Statique et Dynamique de l'Environnement) : identification et structuration des données caractérisant l'environnement des agents.
- MIUS (Modélisation des Interactions Utilisateurs-Systèmes) : simulation et définition des différentes interfaces homme-machine dans chaque scénario (technique de représentation des connaissances sous la forme de situations préétablies).

La phase de conception de MASB est composée de :

- ASMA (Architecture SMA et caractérisation des scénarios) : sélection des scénarios qui vont être implémentés et les rôles des agents dans ces scénarios.
- MO (Modélisation Objet) : raffinement de DSDE en définissant la hiérarchie des objets, leurs attributs et procédures;
- MA (Modélisation Agent informatique) : spécification et modélisation des structures de croyances, des espaces de décisions et des espaces d'action des agents. Un espace de décisions est une structure composée de la hiérarchie des objectifs et des rapports entre ces objectifs et les croyances. Un espace d'action est constitué des plans d'action d'un agent pour atteindre ses objectifs.
- MC (Modélisation de conversations) : spécification des conversations entre agents du système;
- VCSC (Validation de la Conception du Système Complet) : simulation de l'application.

Le découpage du système en niveaux d'abstractions est marqué dans MAS par les concepts de rôles, de scénarios, d'objectifs et d'agents. Le formalisme utilisé ici repose intégralement sur SMAUL2. MASB traite les données à travers les modèles DS, MCD, DSDE, MO et MA; les traitements et les activités à travers DFR et MA; la dynamique à travers MIUS, MC et MA. L'ordre de dérivation des modèles n'est pas clairement spécifié dans MASB. Mais il nous semble que DS dérive des besoins de l'utilisateur; DFR et DSDE dérivent de DS; MCD et MIUS dérivent de DS et DFR; ASMA dérive de DS; MO dérive de DSDE; MA dérive de DS, DFR, MCD; et MC dérive de MA et MIUS. Il nous semble qu'il y a des redondances dans le séquencement de ces modèles qui sont à notre avis complexes (le nombre d'interrelations entre

les modèles étant 13) et incomplets (car les structures et les règles organisationnelles définies dans [ZAM00] ne sont pas explicitement traitées).

MAS-CommonKADS

MAS-CommonKADS contient 7 modèles qui sont :

- Modèle d'Agent (MA) : définit les principales caractéristiques des agents telles que leur capacité de raisonnement, leurs capteurs et émetteurs, les services et la hiérarchie de classes;
- Modèle de Tâche (MT) : décrit les tâches (objectifs) des agents, la décomposition de ces tâches, les méthodes de résolution de problèmes;
- Modèle d'Expertise (ME) : décrit les connaissances sur le domaine d'application des agents (connaissances nécessaires aux agents pour réaliser leurs tâches, leurs comportements proactifs) et de l'environnement (connaissances sur le monde et les autres agents);
- Modèle d'Organisation (MO) : décrit l'organisation qui supportera le SMA et l'organisation sociale de la société d'agents. Cette description utilise une extension du modèle d'objet de OMT et dépend du domaine d'application;
- Modèle de Coordination (MCo) : décrit les conversations entre agents (interaction, protocoles, aptitude à communiquer). Les interactions sont modélisées en utilisant MSC96 [EKK96], HMSC [IUT94] et SDL [ITU94];
- Modèle de Communication (MC) : détaille les interactions hommes-agents informatiques et les facteurs humains pour le développement des interfaces utilisateurs;
- Modèle de Conception (MCn) : met ensemble les modèles précédents et les subdivise en trois sous-modèles :
    - conception du réseau : conception des aspects pertinents des infrastructures du réseau des agents (exigences du réseau, connaissances et facilités télématiques).

- conception d'agents : composition ou décomposition des agents issus de l'analyse selon des critères pragmatiques (les contraintes spécifiques au domaine d'application) et sélection d'architecture convenable pour chaque agent.
- conception de plate-forme : sélection de plate-forme de développement de chaque architecture d'agent.

Le découpage du système se fait autour des concepts de rôles, de tâches, d'agents. Le modèle d'organisation permet d'étudier les stratégies à long terme et les tactiques du système.

MAS-CommonKADS traite les données à travers les modèles MA, ME, MO et MCo; les traitements et les activités à travers MA, MT, et ME; la dynamique du système à travers MCo, MC, MCn.

L'ordre de dérivation des modèles n'est pas clairement indiqué dans cette méthodologie mais il nous semble que : MA dérive de l'analyse du domaine d'application (phase de conceptualisation) et est raffiné par ME et MT; MO dérive de MA et du domaine d'application; MT dérive de MA, ME dérive de MA et de MT, MCo dérive de ME, MC dérive de MA et de MCo, MCn dérive des 6 autres modèles. Chaque modèle est accompagné d'une documentation textuelle détaillée.

Il nous semble, au regard de ces modèles et de leurs descriptions, qu'ils sont plus complets que ceux des autres méthodologies. Cependant, on note une sorte de redondance entre les modèles. Le modèle de conception est particulièrement complexe.

AOMEM

La méthodologie AOMEM est composée de 4 modèles :

- Le Modèle Fonctionnel (MF) décrit les fonctions du système en utilisant les modèles fonctionnels IDEF0 de la méthode IDEF ([BRA85a] [BRA85b]). IDEF est une méthode de modélisation de la gestion de procédures d'entreprise. Un IDEF0 est établit pour chaque fonction. Une fonction dans IDEF correspond à un cas d'utilisation dans OOSE et aux objectifs et plans d'un agent dans AOMEM.

- Le Modèle des Cas d'Utilisation (MCU) décrit les acteurs impliqués dans chaque fonction en utilisant la technique des cas d'utilisation de OOSE. Un acteur dans OOSE correspond à une ressource dans IDEF et à un agent dans AOMEN.
- Le Modèle Dynamique (MD) analyse les interactions entre objets pour chaque cas d'utilisation. Il utilise les diagrammes de trace d'événements pour représenter ces interactions. Les objets passifs représentent les bases de croyances des agents [GEO90]. Le projet Ontolingua [GRU92] a produit un système informatique qui permet de translater les représentations des bases de croyances en base de connaissances, et vice versa. Les objets ayant un comportement dynamique peuvent être des capteurs et des émetteurs des agents [RAO95].
- Le Système Orienté Agent (SOA) identifie les agents du système, détermine leurs objectifs et plans d'actions, leurs croyances, leurs capteurs et émetteurs. Un agent possède des croyances, des objectifs, des plans. Les agents apparaissent dans les cas d'utilisation pour remplacer ou assister les acteurs. SOA identifie aussi les protocoles de coordination des agents en utilisant les diagrammes d'états-transitions.

Le découpage du système utilise les concepts de cas d'utilisation, d'acteur, d'objectif, de comportement et de trace d'événement.

Au niveau du formalisme, AOMEN traite les données (les agents, les objets par exemples) à travers les modèles MCU, MD, MF et SOA; les traitements et les activités à travers MF et SOA; la dynamique du système à travers MD et SOA.

L'ordre de dérivation des modèles est clair mais reste implicite. Ainsi, MF dérive de l'analyse des besoins de l'utilisateur; MCU dérive de MF; MD dérive de MCU et de MF et SOA dérive de MF, MCU et de MD.

Il nous semble que les modèles de AOMEN sont cohérents, moins complexes (le nombre d'interrelations entre modèles étant 6) mais pas complets car ils ne prennent pas en compte véritablement les structures et les règles organisationnelles (décrites formellement dans [ZAM00]) qui supportent les SMA. Les structures de coordination et de connaissances ne sont pas bien développées.

Cassiopée

Cassiopée contient trois principales étapes :

- La première étape Elementary Behaviors (EB) consiste à déterminer les tâches (comportements) élémentaires qui sont nécessaires à l'accomplissement de l'objectif global du système. Cette étape conduit à l'identification des types d'agents.
- La deuxième étape Relational Behaviors (RB) analyse la structure organisationnelle basée sur les dépendances fonctionnelles des tâches élémentaires identifiées à l'étape 1. Cette étape utilise un graphe de dépendances ou de couplage (encore appelé graphe d'influences). Un graphe d'influence permet de déterminer les dépendances pertinentes (appelées ici influences) entre les types d'agents. Les chemins et les nœuds (types d'agents) de ce graphe fournissent une représentation globale de la structure organisationnelle des agents. Ce graphe est une sorte de diagramme de Venn décrivant les relations de dépendance dans l'ensemble des tâches élémentaires.
- La troisième étape Organisational Behaviors (OB) s'intéresse à la dynamique de l'organisation des agents du système. Elle consiste à spécifier les compétences des agents liées à la gestion de formation, de durabilité et de dissolution de groupes. Voir [COL95] pour plus de détails sur les méthodes de Cassiopea.

Le découpage du système se fait en utilisant les concepts d'influences [SIC95], de groupes et de tâches. Les données, les traitements et les activités du système sont traités à travers les trois étapes (EB, RB, OB) de Cassiopée; la dynamique du système à travers OB. RB dérive de EB qui est dérivé de l'analyse des besoins de l'utilisateur. L'étape OB dérive de RB. Bien que ces étapes soient cohérentes, nous ne pourrions affirmer si les modèles qui en découlent sont complets ou complexes.

B) Discussion

Toutes les méthodes ici étudiées possèdent des moyens d'assemblage pour maîtriser la complexité des systèmes qu'elles peuvent concevoir. Cependant, l'étude de la stratégie et de la tactique de ces systèmes n'a été prise en compte que par la méthode MAS-CommonKADS. Les formalismes utilisés (aussi bien au niveau de la spécification qu'au niveau de la conception) par ces méthodes pour la représentation des données, des traitements, des activités et de la

dynamique des systèmes constituent une extension des techniques orientées objet ou des techniques des méthodes à base de connaissance. Ces formalismes diffèrent d'une méthodologie à une autre (voir le tableau 7.1). Ces formalismes et le séquencement des modèles indiquent comment sont traités (représentés) les données, les traitements, les activités et la dynamique des systèmes dans chacune des méthodologies étudiées. Les techniques (UCMs) utilisées par la méthode HLIM pour la représentation sont particulièrement intéressantes (à notre avis) pour la compréhension, l'interprétation et la communication entre les différents acteurs du développement de logiciels. Le formalisme utilisé par la méthode MaSE permet la réutilisabilité automatique des composantes et rend facile la vérification (au niveau de la spécification et au niveau de la conception). L'ordre de dérivation est clairement indiqué dans les méthodes GAIA, MaSE, HLIM, CoMoMAS; implicitement indiqué dans les méthodes AOMEN et Cassiopée et reste flou dans MMTS, MASB, MAS-CommonKADS.

Les modèles de chacune de ces méthodologies ne semblent pas recouvrir toutes les dimensions des SMA et restent donc incomplets. Mais il nous semble, au regard des modèles de MAS-CommonKADS et de leurs descriptions, qu'ils sont plus complets que ceux des autres méthodologies. Les modèles des méthodes MAS-CommonKADS, MASB, MMTS et CoMoMAS sont à notre avis particulièrement complexes.

7.1.3 Dimension agent

Le tableau 7.3 donne la nature, et les caractéristiques des agents du système auquel la méthodologie s'applique.

Critères	Valeurs des critères	MÉTHODOLOGIES								
		GAIA	MaSE	MMTS	HLIM	CoMoMAS	MASB	MAS-Common KADS	AOMEM	Cassiopée
Nature des agents	homogène	non	non	non	non	non	non	non	non	non
	hétérogène	oui	possible	oui	oui	oui	oui	oui	oui	oui
Type d'agents	agents intelligents	possible	possible	oui	oui	oui	oui	oui	oui	oui
	agents interfaces	possible	possible	oui	oui	oui	oui	oui	oui	
	agents mobiles	possible	possible	oui	possible	oui	possible	oui		oui
	agents d'information	possible	oui	oui	oui	possible	oui	oui	oui	possible
	agents autonomes	possible	possible	oui	possible	oui	oui	oui	oui	oui
Attributs des agents	adaptabilité	possible	possible	possible	possible	oui	possible	possible	possible	possible
	autonomie	possible	possible	oui	oui	oui	oui	oui	oui	oui
	comportement coopératif	possible	possible	oui	oui	oui	oui	oui	oui	oui
	capacité déductive	possible	possible	oui	oui	oui	oui	oui	oui	oui
	habileté de communication	possible	possible	oui	oui	oui	oui	oui	oui	possible
	mobilité									
	personnalité	possible	possible	oui	oui	oui	oui	oui	oui	oui
	réactivité	possible	possible	oui	oui	oui	oui	oui	oui	oui
	continuité temporelle	possible	possible	non			non			oui
	comportement délibératif	possible	possible				non	oui	possible	possible
Attributions des agents	stratégie physique	non	non		non	non	oui	possible		possible
	stratégie de l'entité	oui	oui	oui	oui	oui	oui	oui	oui	oui
	stratégie intentionnelle	possible		oui	possible	oui	oui	oui	oui	oui

Tableau 7.3 : Caractéristiques générales des agents du système

Légende

Oui : signifie que la méthodologie a pris en compte cette valeur.
Non : signifie que la méthodologie n'a pas pris en compte cette valeur.
Blanc : signifie qu'on ne peut rien conclure sur la base des documents que nous avons lus.
Possible : signifie qu'au regard des éléments décelés dans les documents, on peut déduire que la méthodologie pourrait prendre en compte cette valeur.

A) Commentaires

GAIA

La méthodologie GAIA est une approche générale et peut convenablement être spécialisée à des architectures spécifiques d'agents. Les agents peuvent être hétérogènes, c'est-à-dire qu'ils peuvent être implémentés dans différents langages de programmation et techniques de communication. Le modèle de rôle est un modèle abstrait d'agents, à partir duquel les types d'agents et les instances sont identifiés. Nous pouvons donc penser que GAIA peut s'appliquer aux systèmes utilisant n'importe quel type d'agent ayant n'importe quels attributs. À travers les permissions (les droits qui légitiment les responsabilités d'un agent) qui sont une représentation d'une base de connaissances du monde réel sur l'agent, on peut dire que les prédictions sont basées sur les hypothèses de l'environnement. Les protocoles d'interactions sont fonction des entrées/sorties des agents qui y sont engagés. Cependant, il faut noter que les travaux qui sous-tendent cette méthodologie restent encore théoriques et n'ont pas encore connu une phase pratique [WOO00b].

MaSE

MaSE est une méthodologie générale, en ce sens qu'elle peut s'appliquer à des domaines variés. Les diagrammes d'agents sont génériques et ne permettent donc pas d'indiquer la nature, et les attributs des agents. Les types d'agents sont fonction du domaine d'application à modéliser. La définition de l'architecture des agents utilise le langage AgDL qui est basé sur la logique des prédicats du premier ordre. Nous pouvons donc dire qu'un agent peut avoir des compétences citées dans le tableau 7.3. Les protocoles d'interactions sont fonction des entrées/sorties des agents qui y sont engagés (il faut noter l'utilisation des machines d'états finis des DCC). Nous ne pouvons rien conclure sur la présence dans les modèles des mécanismes d'attribution d'intentions des agents.

MMTS

La méthodologie MMTS peut utiliser des agents hétérogènes. Ces agents sont des agents intelligents et peuvent être des agents personnels, agents mobiles, des agents d'information et des agents autonomes. Le système de gestion du trafic aérien que les auteurs de MMTS ont implémenté avec cette méthodologie est un agent interface. Les agents peuvent être adaptables,

c'est-à-dire peuvent avoir une habileté à apprendre et à s'améliorer avec l'expérience. Ils ont un comportement coopératif c'est-à-dire qu'ils sont capables de travailler en équipe pour atteindre un objectif commun. Ils sont dotés d'une capacité de raisonnement et ont une habileté de communication (utilisent les actes de discours comme langage de communication entre agents). À travers les modèles de croyance et de plan, on note que les agents peuvent être réactifs c'est-à-dire agir selon ce qu'ils perçoivent. Les agents peuvent parfois agir comme un humain car, avec la technologie de MMTS, les auteurs ont fait la simulation du combat de l'air. À travers les modèles d'interactions et de croyances on peut noter que les attributions des agents sont basées sur les entrées/sorties et sur les hypothèses de l'environnement.

HLIM

La méthodologie HLIM peut supporter le développement des agents hétérogènes, des agents intelligents, des agents interfaces, des agents mobiles et des agents d'information. La présence des diagrammes de dépendance et de juridiction nous fait douter, cependant, de l'autonomie des agents. Ces agents peuvent avoir un comportement coopératif (les chemins des UCMs impliquent le comportement coopératif de multiples agents), une capacité déductive, une habileté de communication avec l'utilisation des actes de discours. Ils peuvent être capables de faire un choix et d'agir en fonction de ce choix. À travers les modèles utilisés (Internal Agent Model par exemple) dans HLIM, on peut remarquer que les attributions des agents sont basées sur leurs entrées/sorties et leurs croyances sur l'environnement.

CoMoMAS

Les agents de CoMoMAS peuvent être hétérogènes. Ce sont des agents autonomes, intelligents, mobiles, interfaces ou d'information. Ils possèdent presque tous les attributs que nous avons cités dans la grille de cette dimension. Il faut noter l'existence dans CoMoMAS d'une étape d'analyse des compétences des agents du système; le modèle d'expertise en est une justification. CoMoMAS est avant tout une méthodologie à base de connaissances formulées à partir des hypothèses, des croyances émises sur l'environnement. Les attributions des agents, qui sont prises en compte dans le modèle d'expertise, sont donc basées sur la stratégie intentionnelle et aussi sur la stratégie de l'entité.

MASB

La méthodologie MASB peut utiliser des agents hétérogènes. Ces agents sont des agents intelligents, interfaces, d'information, autonomes et peuvent être des agents mobiles. Ils possèdent presque tous les types de compétence spécifiés dans cette dimension. Ils sont des agents BDI et des agents humains. Selon les auteurs de MASB, ils peuvent évoluer de façon indépendante suivant les lois qui régissent l'environnement (par exemple, les lois gravitationnelles, les lois du marché ou les lois sociales). Ce qui signifie donc que MASB peut représenter les agents selon les trois stratégies de Dennett (voir section 6.2.3). C'est le modèle d'agent qui contient des mécanismes de reconnaissances et d'attributions d'intentions des agents.

MAS-CommonKADS

Les agents des SMA que la méthodologie MAS-CommonKADS peut implémenter peuvent être hétérogènes et sont de n'importe quel type de cette dimension. Ces agents peuvent avoir n'importe quel attribut de la grille (sauf continuité temporelle sur laquelle on ne peut rien dire) à cause du modèle d'expertise. Leurs attributions sont basées sur la stratégie de l'entité, la stratégie intentionnelle et peuvent être basées sur la stratégie physique à cause des modèles d'organisation et d'expertise. C'est le modèle d'expertise qui contient des mécanismes de reconnaissance et d'attribution d'intentions des agents.

AOMEM

La méthodologie AOMEM peut modéliser les agents BDI, interfaces, autonomes et d'information. Ces agents peuvent être hétérogènes et possèdent presque tous les compétences citées dans cette dimension. De part les bases de croyances et l'intelligence des agents, leurs attributions peuvent être basées sur la stratégie de l'entité et la stratégie intentionnelle.

Cassiopée

Les agents qu'on peut concevoir avec Cassiopée sont de types intelligents, mobiles et autonomes. Ces agents ont des compétences (attributs) d'adaptabilité, d'autonomie (semi-autonomie), de déduction, de coopération, de compétition, de réactivité, de personnalité, et de continuité temporelle. Un agent est à l'écoute des autres, c'est- à dire qu'il tient compte des autres avant d'agir. Étant donné que Cassiopée est destinée aux robots footballeurs, nous en déduisons que les attributions des agents peuvent être représentées suivant les trois stratégies de

Dennett. En effet, un agent tient compte des caractéristiques de l'environnement, des signes reçus et produits par les autres et leur attribue des intentions. La dynamique des dépendances entre les rôles inclut la reconnaissance des rôles des autres. L'attribution d'intention aux autres agents se note à travers les rôles relationnels et organisationnels.

B) Discussion

Toutes les méthodologies étudiées peuvent permettre la conception des agents hétérogènes dans un SMA. Ces agents peuvent être de n'importe quel type et avoir n'importe quel attribut (à part la mobilité, la continuité temporelle et le comportement délibératif) cité dans cette dimension. Cependant GAIA et MaSE, étant des approches générales, elles ne permettent pas d'identifier clairement les compétences que les agents peuvent avoir. MAS-CommonKADS, à travers son modèle d'expertise, prend en compte le comportement délibératif des agents. À part GAIA et MaSE, toutes ces méthodologies peuvent représenter les attributions des agents suivant la stratégie de l'entité et la stratégie intentionnelle (voir section 6.2.3). Seule MASB semble prendre en compte la stratégie physique. MAS-CommonKADS et MASB semblent avoir pris en compte presque tous les critères de cette dimension agent.

7.1.4 Dimension organisation

Le tableau 4 donne une vue globale sur les types de système que la méthodologie peut représenter, ainsi que les caractéristiques de l'environnement sur lequel les agents du système agissent.

Critères	Valeurs des critères	MÉTHODOLOGIES								
		GAIA	MaSE	MMTS	HLIM	CoMoMAS	MASB	MAS-Common KADS	AOMEM	Cassiopée
Image d'organisation	systèmes hiérarchiques			oui	oui	possible	possible	possible		possible
	systèmes distribués	oui	oui	oui	oui	oui	oui	oui	oui	oui
	systèmes ouverts	non	non	possible	possible	possible	non	non	non	non
	systèmes holoniques			possible	possible	possible	possible	possible		possible
Nature de l'environnement	structuré	oui	oui	oui	oui	oui	oui	oui	oui	oui
	stable	non	non	oui	oui		possible	possible		possible
	déterministe									
	explicite	oui	non	oui	oui	oui	oui	oui	oui	possible
	observable									
Type de l'environnement	Actif	oui	oui	oui	oui	oui	oui	oui	oui	oui
	passif	oui	oui	oui	oui	oui	oui	oui	oui	oui
Caractéristiques des données traitées dans l'organisation	qualitative	oui	oui	oui	oui	oui	oui	oui	oui	oui
	quantitative	oui	oui	oui	oui	oui	oui	oui	oui	oui

Tableau 7.4 : Caractéristiques de l'organisation et de l'environnement du système

<u>Légende</u>

Oui : signifie que la méthodologie a pris en compte cette valeur.

Non : signifie que la méthodologie n'a pas pris en compte cette valeur.

Blanc : signifie qu'on ne peut rien conclure sur la base des documents que nous avons lus.

Possible : signifie qu'au regard des éléments décelés dans les documents, on peut déduire que la méthodologie pourrait prendre en compte cette valeur.

A) Commentaires

GAIA

La méthodologie GAIA peut représenter les systèmes distribués dont l'environnement est structuré (à travers les modèles de rôle et de service). Par contre, GAIA ne peut pas s'appliquer

aux systèmes stables et ouverts. En effet, GAIA ne prend pas en compte la modélisation de la création et de la suppression des rôles. Les auteurs de GAIA prévoient dans l'avenir étendre la méthode aux systèmes ouverts. L'environnement est explicite à travers les permissions des rôles et, en principe, actif et passif. Mais nous ne pouvons rien conclure sur l'observabilité et le déterminisme de cet environnement (voir section 6.2.4). Les données que GAIA peut traiter peuvent être qualitatives et quantitatives. L'aspect des structures organisationnelles (définies dans [ZAM00]) dans l'analyse et la conception des systèmes agents demande assez d'effort. De telles structures sont implicitement définies dans la méthodologie GAIA à travers les modèles de rôle et d'interaction. Cependant, la représentation explicite de ces structures qui est utile pour comprendre les structures organisationnelles des systèmes n'existe pas dans GAIA. Les auteurs ont conçu des modèles d'organisation, qui sont jusque-là théoriques, et pensent les intégrer à GAIA [ZAM00].

MaSE

Les diagrammes de déploiement permettent de définir complètement l'architecture du système multi-agents. La méthodologie MaSE peut représenter, en principe, les systèmes distribués. Mais elle n'est sans doute pas destinée aux systèmes ouverts, car elle ne traite pas de l'intégration et de la suppression dynamique des agents. L'identification des types d'agents est basée sur l'identification des rôles, ce qui indique que l'environnement sur lequel les agents agissent est structuré. Cet environnement n'a pas été explicitement formulé par MaSE. MaSE ne peut pas s'appliquer aux systèmes stables car, ils ne peuvent pas résister aux pertubations éventuelles de l'environnement. Mais nous ne pouvons rien conclure sur l'observabilité et le déterminisme de cet environnement qui est en principe actif et passif. MaSE peut traiter aussi bien des données qualitatives que quantitatives.

MMTS

La méthodologie MMTS peut implémenter les systèmes hiérarchiques, distribués, holoniques et ouverts. En effet, MMTS permet de représenter les responsabilités de la création et de la suppression d'agent, et de la formation d'équipes. Cependant, les techniques de modélisation de ces rapports entre agents n'ont pas été traitées dans cette méthodologie ou, du moins, dans les documents que nous avons lus. Le modèle d'agent nous permet de dire que l'environnement sur lequel les agents agissent est structuré car les rôles et les fonctions sont

clairement définis. Des plans pour de nouveaux contextes peuvent être ajoutés sans changer les plans déjà existants pour la réalisation du même objectif. Le changement de comportement ne change pas les objectifs. Ce qui nous fait dire que l'environnement peut être stable, c'est-à-dire faire face aux perturbations. Cet environnement est explicite à travers les modèles de croyance, d'objectif et de plan. Ainsi, les agents perçoivent clairement les données et leurs structures et donc pourront agir conséquemment. Nous ne pouvons pas affirmer que cet environnement est déterministe et observable. Il est actif et passif. Les données traitées par cette méthodologie peuvent être qualitatives (les croyances et les objectifs des agents par exemple), mais aussi quantitatives (les types d'agents et leurs instances).

HLIM

La méthodologie HLIM peut représenter les systèmes hiérarchiques, distribués et donc holoniques. Par l'utilisation des notions de «slots» (conteneurs des composantes dynamiques déjà opérationnelles) et des «pools» (conteneurs des composantes qui ne sont pas en exécution) des UCMs. L'environnement sur lequel les agents agissent est structuré car les rôles et les fonctions sont clairement définis. Il est stable par la décomposition orientée objectif. Cet environnement est explicite à travers l'utilisation des modèles de HLIM, notamment les modèles IAM et CTM. Mais nous ne pouvons rien conclure sur l'observabilité et le déterminisme de cet environnement qui est actif et passif. Les données traitées dans les systèmes que HLIM peut supporter sont aussi bien qualitatives que quantitatives.

CoMoMAS

CoMoMAS peut représenter les systèmes distribués. Les méthodes de coopération et de réorganisation dans le langage CML nous font penser que CoMoMAS pourrait permettre de représenter les systèmes ouverts. Son environnement est bien structuré, explicite, actif et passif. Le modèle du système définit les structures organisationnelles de la société que forment les agents. Le déterminisme et l'observabilité de cet environnement ne sont pas traités dans CoMoMAS. Les données utilisées dans CoMoMAS sont aussi bien quantitatives que qualitatives.

MASB

MASB est destinée aux systèmes distribués, mais pas aux systèmes ouverts car tous les scénarios sont prédéfinis. L'environnement dans lequel les agents évoluent est bien structuré et explicite. Mais nous ne pouvons rien conclure sur l'observabilité et le déterminisme de cet environnement qui est actif et passif. Les données traitées dans MASB sont quantitatives et qualitatives.

MAS-CommonKADS

Le modèle d'organisation de MAS-CommonKADS est générique et dépend du domaine d'application. Ce modèle peut représenter les systèmes hiérarchiques, distribués et holoniques mais pas les systèmes ouverts car tous les scénarios sont prédéfinis. L'environnement des agents est bien structuré, explicite, stable, actif et passif. Les données traitées par cette méthodologie peuvent être quantitatives et qualitatives.

AOMEM

Nous n'avons pas remarqué dans la documentation (que nous avons lue), décrivant cette méthodologie, les structures organisationnelles. Cependant, nous pouvons dire qu'elle peut modéliser les systèmes distribués et que l'environnement sur lequel les agents agissent est structuré (les fonctions du système étant bien identifiées), explicite (à cause des bases de croyances), actif et passif. Les données traitées dans AOMEN sont quantitatives et qualitatives.

Cassiopée

Les deux dernières étapes RB, OB de Cassiopée traitent de l'organisation statique et dynamique de la société d'agents.

Cassiopée peut représenter les types d'organisation hiérarchiques, distribués et holoniques ; mais elle n'est certainement pas destinée aux systèmes ouverts car elle ne traite pas de la création et de la suppression dynamiques des agents. L'environnement des agents est bien structuré de part l'identification des tâches élémentaires et peut être explicite. Les données utilisées dans Cassiopée sont aussi bien quantitatives que les qualitatives.

B) Discussion

Toutes les méthodes sont en principe destinées aux systèmes distribués car les SMA sont des applications distribuées. Aucune d'elles ne peut véritablement concevoir un système ouvert. Les SMA ouverts constituent actuellement un défi à relever pour les chercheurs. Les structures organisationnelles ne sont pas clairement explicitées dans ces méthodes à part Cassiopée (dont le mot-clé est l'organisation) et MAS-CommonKADS (à travers son modèle d'organisation). En effet, les modèles relationnels et organisationnels de Cassiopée traitent de l'organisation statique et dynamique de la société d'agents. Pour chacune de ces méthodes, l'environnement est structuré de par leurs modèles de rôle et d'agent. Mais nous ne pouvons rien conclure sur l'observabilité et le déterminisme de cet environnement qui est actif et passif. L'environnement dans les méthodes MMTS et HLIM peut être stable par la décomposition orientée objectif. Dans MMTS, des plans pour de nouveaux contextes peuvent être ajoutés sans changer les plans déjà existants. Cela se remarque dans les modèles de plans et d'objectifs. Les données traitées par ces méthodologies sont aussi bien quantitatives que symboliques (les croyances des agents par exemples).

7.1.5 Dimension coopération

Le tableau 7.5 indique les notions de coopération utilisées dans le système que la méthodologie peut représenter.

<u>Légende</u>

Oui : signifie que la méthodologie a pris en compte cette valeur.

Non : signifie que la méthodologie n'a pas pris en compte cette valeur.

Blanc : signifie qu'on ne peut rien conclure sur la base des documents que nous avons lus.

Possible : signifie qu'au regard des éléments décelés dans les documents, on peut déduire que la méthodologie pourrait prendre en compte cette valeur.

Critères	Valeurs des critères	GAIA	MaSE	MMTS	HLIM	CoMoMAS	MASB	MAS-Common KADS	AOMEM	Cassiopée
		MÉTHODOLOGIES								
Types de communication	communication entre agents hétérogènes	oui	oui	oui	oui	oui	oui	oui	oui	oui
	communication agents-humains	oui	oui	oui	oui	oui	oui	oui	oui	oui
Modes de communication	direct		oui	oui	oui	oui	oui	oui	oui	oui
	indirect		non		oui	non			non	oui
	synchrone		possible	possible	oui	oui	possible	oui	possible	oui
	asynchrone		possible	possible	oui	oui	possible	oui	oui	oui
Langage de communication	signaux	possible	possible	oui	oui	oui	oui	oui	oui	oui
	actes de discours		possible	oui	oui	oui	oui	oui	oui	non
	autres	non	non	non	non	non	non	non	non	non
Modèle de coopération	négociation		oui	oui	oui	oui	oui	oui	oui	oui
	délégation de tâches		possible	oui	oui	oui	oui	oui	possible	oui
	planification multi-agents		possible	oui	oui	oui	oui	oui	oui	oui
Type de contrôle	centralisé					possible		possible	non	non
	hiérarchique				oui	possible	possible	oui	possible	possible
	distribué		possible	possible	possible	possible	oui	oui	oui	oui
Interaction	statique	oui	oui	oui	oui	oui	oui	oui	oui	oui
	dynamique			oui	oui	possible	oui	oui	possible	oui
	moteur d'interaction distribué	possible	oui	oui			oui	possible		possible
	moteur d'interaction centralisé		non				oui	oui		
	protocoles d'interaction explicites	oui	oui	oui	oui	oui	oui	oui	oui	oui
	protocoles d'interaction implicites	non	non	non	non	non	non	non	non	non
	mécanisme d'interaction résout les états non coopératifs	oui	non	possible	oui	oui		oui	non	oui

Tableau 7.5 : Les concepts coopératifs utilisés par la méthodologie

A) Commentaires

GAIA

Dans la méthodologie GAIA, la coopération inter-agents est quelque peu appauvrie. GAIA peut modéliser aussi bien la communication entre agents hétérogènes que la communication agents-humains. GAIA n'a pas été conçue pour un mode de communication, ni pour un langage de communication, particuliers; mais d'après ses auteurs, on peut l'adapter à un type particulier de mode et de langage de communication. Il faut noter que les modèles relationnels de GAIA définissent les liens de communication qui existent entre les types d'agents, mais pas les messages qui sont échangés, ni quand ces messages sont envoyés. La nature du modèle de coopération n'est pas clairement définie dans GAIA et donc les concepts de coopération utilisés dans les modèles d'interaction ne sont pas clairement définis. Le type de coordination utilisé dans les modèles d'interaction dépend de la nature des rôles et de la structure organisationnelle. Or, cette structure reste implicite dans GAIA. Dans l'état actuel des travaux des auteurs de GAIA, on ne peut donc pas indiquer à quels types de contrôle GAIA peut s'appliquer. L'interaction est tout au moins statique. Le moteur d'interaction est distribué, c'est-à-dire interne à chaque agent, puisque à chaque rôle correspond un modèle d'interaction. Les protocoles d'interaction sont explicites et sont supportés par la notation de Fusion [COL94]. Grâce aux propriétés "safeties" des responsabilités des rôles, on peut déduire que le mécanisme d'interaction permet de résoudre des petits conflits qui apparaissent lors de l'exécution du système.

MaSE

La méthodologie MaSE devrait pouvoir modéliser la communication entre agents hétérogènes, car ses modèles de communication sont génériques et ne précisent pas la nature et les caractéristiques des langages de communication. Ces langages de communication peuvent être basés sur les signaux et/ou sur les actes de discours. Le mode de communication peut être direct (utilisation de message) et devrait être synchrone et asynchrone par l'utilisation des MEF. MaSE peut modéliser les systèmes distribués et, par conséquent, le type de coordination utilisé dans les modèles d'interaction peut être distribué. Les concepts de coopération utilisés dans les modèles d'interaction de MaSE peuvent être la négociation, la planification multi-agents et la délégation de tâches. L'interaction est statique (utilisation de message), mais rien dans le

document de la méthodologie que nous avons lu ne nous permet de dire qu'elle est dynamique. Le moteur d'interaction peut être distribué, c'est-à-dire interne à chaque agent (chacun de ces agents étant distribués). Les protocoles d'interaction sont explicités par les machines d'états finis dans les diagrammes de classes de communication de la méthodologie MaSE. Bien que la résolution de conflits entre agents constitue un défi majeur pour l'auteur de cette méthodologie, nous ne pouvons pas affirmer que les mécanismes d'interaction utilisés dans MaSE permettent de résoudre des états non coopératifs.

MMTS

Les modèles d'interaction de la méthodologie MMTS ne mettent pas l'accent sur un type particulier d'interaction. Ils sont génériques et dépendent du domaine d'application. La méthodologie MMTS peut modéliser la communication entre agents hétérogènes et la communication entre agents et humains. Le mode de communication de MMTS peut être direct, synchrone (des messages par téléphone) et asynchrone. Les langages de communication sont surtout basés sur les actes de discours (les actes de discours indiquent toutes les actions intentionnelles effectuées au cours d'une communication). Les concepts utilisés dans le modèle de coopération peuvent être la négociation, la délégation de tâche et la planification multi-agents. Les modèles d'interaction capturent aussi le contrôle des rapports entre les classes d'agents, mais le type de contrôle (qui pourrait être distribué) est fonction du domaine d'application. L'interaction peut être statique (envoie de messages) mais aussi dynamique à travers le modèle de plan. D'après les auteurs de MMTS, les rapports entre agents, tels que les responsabilités pour la création, la suppression d'agent, et la formation d'équipe, sont pris en compte par leur méthodologie. Mais les techniques de modélisation de ces rapports n'ont pas été traitées dans les documents présentant cette méthodologie que nous avons lus. Le moteur d'interaction peut être interne à chaque agent. À travers les modèles de plan, on note que les protocoles d'interaction sont explicites et que le mécanisme d'interaction peut résoudre quelques types de conflits à cause de l'expression des conditions sur les transitions entre états.

HLIM

La méthodologie HLIM peut modéliser la communication entre agents hétérogènes et la communication entre agents et humains. Le mode de communication peut être direct, indirect et synchrone (HLIM ayant été appliquée à un système de téléphonie). Les langages de

communication utilisés par les agents (à travers le modèle CtM) peuvent être basés sur les signaux et sur les actes de discours. Les concepts de coopération utilisés dans HLIM sont les concepts de délégation de tâches, de négociation et les concepts de planification multi-agents. Le diagramme de juridiction du modèle ARM nous permet d'affirmer que la coordination est de type hiérarchique, c'est-à-dire ramenée à une direction dans laquelle il y a une série ascendante de pouvoirs ou de décisions. L'interaction est statique et peut être dynamique compte tenu de la paramétrisation des conversations entre agents dans les DHC. La documentation lue sur la méthodologie ne nous permet pas de voir si le moteur d'interaction peut être distribué ou centralisé. Les protocoles d'interaction sont explicites par l'utilisation des chemins des UCMs. Le diagramme de juridiction et le modèle CtM permettent la définition d'un mécanisme d'interaction capable de résoudre des conflits.

CoMoMAS

Dans CoMoMAS, la communication peut s'effectuer aussi bien entre agents hétérogènes qu'entre agents humains. Le mode de communication peut être direct, synchrone et asynchrone. Les langages de communication peuvent être basés sur les signaux (présence des agents réactifs) et sur les actes de discours (présence des agents cognitifs). On note l'utilisation dans le modèle de coopération de la méthodologie CoMoMAS des concepts de négociation, de délégation de tâche et de planification multi-agents. Bien que ce modèle détermine les stratégies de coordination et de négociation, nous ne pourrons indiquer le type de coordination utilisé. Nous supposons que ce type dépendra de l'application considérée. L'interaction est statique et pourrait être dynamique à cause des méthodes de réorganisation introduites dans le langage CML. Nous ne pourrions affirmer si le moteur d'interaction est distribué ou centralisé. Le modèle de coopération permet d'identifier les protocoles d'interactions et de résoudre les conflits pouvant apparaître dans les interactions des agents.

MASB

Dans MASB, la communication peut s'effectuer entre agents hétérogènes et entre agents et humains. Le mode de communication peut être direct, synchrone ou asynchrone. Les langages de communication sont basés sur les signaux et les actes de discours. Les concepts de négociation, de délégation de tâche et de planification multi-agents sont remarqués dans le modèle de coopération. Le type de contrôle peut être hiérarchique et distribué. L'interaction est

statique et dynamique. Le moteur d'interaction peut être distribué à cause de la présence des agents de type gestionnaire de scénarios et centralisé à cause de la présence des agents de type serveur d'objets. Les protocoles d'interactions sont explicites à travers les modèles d'agents. Nous ne pouvons pas affirmer que le mécanisme d'interaction permet de résoudre les états non coopératifs.

MAS-CommonKADS

Dans la méthodologie MAS-CommonKADS, la communication peut s'effectuer aussi bien entre agents hétérogènes qu'entre agents humains. Le mode de communication peut être direct (utilisation de messages), synchrones et asynchrone (voir MCo). Les langages de communication entre les agents peuvent être basés sur les signaux (présence des agents réactifs) et sur les actes de discours (présence des agents cognitifs). On peut noter l'utilisation des concepts de négociation, de délégation de tâche et de planification multi-agents dans le modèle de coordination. Le type de contrôle pourrait être hiérarchique et distribué dépendamment du domaine d'application. L'interaction est statique (voir MO) et dynamique (voir MCo). Le moteur d'interaction est centralisé (utilisation de serveur) et peut être distribué. Les protocoles d'interactions sont explicites. Le modèle de coordination offre des possibilités de résolution de conflits.

AOMEM

Dans la méthodologie AOMEM la communication peut s'effectuer entre agents hétérogènes, d'une part, et entre agents et humains, d'autre part. Le mode de communication utilisé par les agents peut être direct, asynchrone et synchrone. Leurs langages de communication peuvent être basés sur les signaux et les actes de discours. Il faut noter l'utilisation possible des langages de communication comme COOL et AgenTalk. La négociation et la planification multi-agents sont les concepts utilisés dans les modèles de coopération. Le type de contrôle peut être hiérarchique et distribué. L'interaction est statique et peut être dynamique à cause de l'utilisation possible de AgenTalk; mais nous ne pouvons indiquer si elle est dynamique et si le moteur d'interaction est distribué ou centralisé. Les protocoles d'interaction sont explicites mais les mécanismes d'interaction ne permettent pas de gérer les états non coopératifs.

Cassiopée

Cassiopée peut permettre la communication entre agents hétérogènes et entre agents-humains. Dépendamment du domaine d'application, le mode de communication (dont le choix est réservé au concepteur) peut être direct (envoie de message), indirect (tableau noir), synchrone et asynchrone. Le langage de communication est principalement basé sur les signaux. Le modèle de coopération peut utiliser les concepts de négociation, de délégation de tâches et de planification multi-agents (voir dimension méthodologie). Le type de contrôle est distribué et peut être hiérarchique. L'interaction est statique (RB) et dynamique (OB). Le moteur d'interaction peut être distribué et centralisé. Les protocoles d'interactions sont sans doute explicites à cause de l'utilisation des techniques dont nous avons parlées dans la dimension méthodologie. Nous pensons que le mécanisme d'interaction (mécanisme de réseau de contrats) pourrait permettre de résoudre les états concurrents [SYC89], d'incompréhension et d'ambiguïté.

B) Discussion

Toutes les méthodologies, ici étudiées, peuvent modéliser les communications entre agents hétérogènes et entre agents et humains. À part GAIA, le mode de communication utilisé par ces méthodes peut être direct, synchrone et asynchrone. Le langage de communication peut être basé sur les signaux et sur les actes de discours (à l'exception de GAIA où ceci n'a pas été indiqué, et de Cassiopée qui utilise seulement les signaux). Les modèles de coopération de ces méthodes à l'exception de GAIA exploitent les mêmes concepts. Le type de contrôle qui est en principe distribué reste implicite dans ces méthodologies à l'exception de HLIM et MAS-CommonKADS où c'est plus ou moins indiqué. L'interaction est statique dans toutes ces méthodes et dynamique dans les méthodes MMTS, HLIM, MASB, MAS-CommonKADS et surtout dans CASSIOPÉE. Toutes ces méthodes ont le mérite d'expliciter les protocoles d'interactions. La résolution des états non coopératifs n'a pas été véritablement prise en compte dans ces méthodes, à part GAIA, HLIM, CoMoMAS, MAS-CommonKADS et CASSIOPEA où le mécanisme d'interaction permet de résoudre de petits conflits. Il est à noter que la coopération n'est pas très développée dans la méthodologie GAIA.

7.1.6 Dimension technologie

Le tableau 7.6 permet d'appréhender le résultat final que l'on peut attendre des différentes méthodes, du point de vue logiciel (les types de logiciels qu'on peut avoir, ainsi que leurs caractéristiques).

		MÉTHODOLOGIES								
Critères	**Valeurs des critères**	**GAIA**	**MaSE**	**MMTS**	**HLIM**	**CoMoMAS**	**MASB**	**MAS-CommonKADS**	**AOMEM**	**Cassiopée**
Mode de traitement	batch	non	non	non	non	non	non	non	non	non
	interactif	possible	possible	oui	oui	oui	oui	oui	oui	oui
	client-serveur	possible		possible		possible	oui	oui		oui
	synchrone	possible	possible	possible	oui	oui	possible	oui	possible	oui
	asynchrone	possible	possible	possible	possible	oui	possible	oui	oui	oui
	distribué	possible	oui	possible	oui	oui	oui	oui	oui	oui
Type d'interface	classique	oui	oui	oui	oui	oui	oui	oui	oui	oui
	adaptable	possible		possible	possible	possible	possible	oui	oui	possible
	adaptative	possible		possible	oui	oui	oui	oui	possible	possible
	assistante	possible		oui	oui	oui	oui	oui	oui	oui
Programmation	structurée	oui	possible	oui	oui	oui	oui	oui	oui	oui
	orientée-objet	oui	possible	oui	oui	oui	oui	oui	oui	oui
	orientée agent			possible						
Type d'application visée	simulation	possible		oui		possible	possible	possible	possible	oui
	résolution de problème	possible	possible	oui	oui	oui	oui	oui	oui	oui
	intégration	possible	oui	possible		possible		possible		oui
Environnement de développement		non	non	non	non	non	non	non	non	non

Tableau 7.6 : Logiciels réalisables par les méthodologies.

<u>Légende</u>

Oui : signifie que la méthodologie a pris en compte cette valeur.

Non : signifie que la méthodologie n'a pas pris en compte cette valeur.

Blanc : signifie qu'on ne peut rien conclure sur la base des documents que nous avons lus.

Possible : signifie qu'au regard des éléments décelés dans les documents, on peut déduire que la méthodologie pourrait prendre en compte cette valeur.

A) Commentaires

GAIA

L'objectif principal de GAIA est de combiner plusieurs entités hétérogènes qui agissent ensemble pour atteindre un seul but. Les auteurs de GAIA croient que leur méthodologie est appropriée au développement des systèmes comme ADEPT [JEN96a] et ARCHON [JEN96b]. Ces systèmes peuvent comporter des agents hétérogènes. Les habiletés des agents de ces systèmes et les services qu'ils fournissent, d'une part, et la structure organisationnelle, d'autre part, sont statiques en ce sens qu'ils ne peuvent pas être modifiés pendant l'exécution de ces systèmes. Les types d'applications visés sont donc la résolution de problèmes et la simulation. Les modes de fonctionnement des logiciels visés ne sont pas typiquement définis et restent génériques. Les types d'interface homme-machine peuvent être, adaptatifs, assistants et flexibles. Il est à noter que l'état actuel de GAIA est embryonnaire et n'a donc connu aucune phase pratique. La programmation peut être structurée et orientée-objet.

MaSE

L'objectif de l'auteur de cette méthodologie est d'intégrer MaSE et AgML à un analyseur automatique de systèmes multi-agents appelé AgentTool. AgentTool permettra de vérifier formellement et de générer, par un interface utilisateur graphique basé sur le langage AgML, les systèmes multi-agents qui sont corrects par construction. Mais la documentation lue sur la méthodologie MaSE ne nous permet pas de classifier les modes de traitement, les types d'interface hommes-machines et le type de programmation que MaSE peut utiliser.

MMTS

Le programme de recherche des auteurs de cette méthodologie est centré sur la conception, l'implémentation et la compréhension théorique d'une architecture particulière des agents BDI. Ils ont déjà eu à appliquer leur technologie aux domaines incluant la gestion du trafic aérien, la simulation de combat aérien et le processus de gestion du commerce. Le mode de traitement des logiciels visés peut être interactif, client-serveur ou distribué. Le type d'interface hommes-machines peut être adaptable (dynamique) et adaptatif (l'interface vise à s'adapter aux besoins de l'utilisateur et se comporter comme un assistant humain). Il faut noter que le système de gestion du trafic aérien conçu avec MMTS est responsable de l'assistance aux contrôleurs

humains, pour déterminer la séquence des atterrissages des avions de plusieurs provenances sur un seul aéroport, afin d'assurer la sécurité sur l'aéroport et les autres contraintes (les contraintes temporelles par exemple) en minimisant les retards et les encombrements. La programmation peut être structurée et orientée-objet. Le type d'application visé peut être la simulation ou la résolution de problèmes.

HLIM

L'un des objectifs des auteurs de cette méthodologie est de faire de HLIM une contribution au développement de la programmation orientée agent. Le mode de traitement peut être interactif et synchrone. Les interfaces hommes-machines peuvent être de type adaptable, adaptative, ou assistante. Les types d'application visés par HLIM sont surtout la résolution de problèmes. Cette méthodologie a été appliquée à la conception d'un système de téléphonie sous forme de prototype [EAL99]. Ainsi, elle fournit des directives qui peuvent aider les ingénieurs télécoms dans leurs tâches.

CoMoMAS

Le mode de traitement des logiciels visés par CoMoMAS peut être interactif, client-serveur, synchrone, asynchrone et distribué. L'interface utilisateur peut être adaptable, adaptative ou assistant à cause de l'utilisation de CLOS qui est un langage de programmation objet à interface graphique. La programmation est structurée et orientée-objet. CoMoMAS est particulièrement destinée à la résolution de problèmes. Elle a été utilisée pour la modélisation de AIbot agent model [20]. AIbot agent model a été utilisé pour une application de surveillance avec plusieurs robots mobiles appelés Nomad200 robots. À cause de l'utilisation de la plate-forme MICE, nous pouvons dire que CoMoMAS peut viser des applications de simulation. Elle a aussi été intégrée à MICE et à KADSTOOL, le support logiciel de la méthodologie CommonKADS dont dérive CoMoMAS.

MASB

Le mode de traitement des logiciels visés par MASB peut être interactif, client-serveur, synchrone, asynchrone et distribué. L'interface homme-machine peut être adaptable, adaptative ou assistant. Un accent est mis sur l'étude des interfaces utilisateurs à travers le modèle d'interaction homme-machine MIUS de la phase d'analyse. La programmation peut être

structurée et orientée-objet. MASB peut concevoir des SMA destinés à la résolution de problèmes. Elle peut également viser des applications de simulation.

MAS-CommonKADS

Le mode de traitement des logiciels visés par la méthodologie MAS-CommonKADS peut être interactif, client-serveur, synchrone, asynchrone et distribué. Le type d'interface homme-machine peut être adaptable, adaptative et assistant. La programmation peut être structurée et orientée-objet. MAS-CommonKADS est destinée à la résolution de problèmes (utilisation des méthodes de résolution de problèmes) et peut être également destinée aux applications de simulation et d'intégration (car les modèles sont génériques).

AOMEM

Le mode de traitement des logiciels visés par la méthodologie AOMEN peut être interactif, synchrone, asynchrone et distribué. L'interface peut être adaptable, adaptative et assistante. La programmation peut être structurée et orientée-objet. Les applications visées peuvent être de type résolution de problème et de type simulation.

Cassiopée

La méthodologie Cassiopée est destinée aux robots footballeurs. Le mode de traitement des logiciels visés par Cassiopée peut être interactif, client-serveur (à cause de l'utilisation des techniques de [PAP90]), synchrone, asynchrone et distribué. Bien que Cassiopée ne fait pas cas des interfaces, nous pouvons penser que les types d'interfaces qu'elle peut permettre sont assistant, adaptable et adaptatif. La programmation est structurée et orientée-objet. Les applications visées par Cassiopée sont de type résolution de problèmes, simulation et intégration.

B) Discussion

Toutes les méthodes sont destinées à la résolution de problèmes. La phase d'implémentation des systèmes de ces méthodes peut utiliser une programmation structurée et orientée-objet. Mais nous ne pouvons pas affirmer si cette programmation peut être orientée agent puisqu'il nous semble que les caractéristiques d'une programmation orientée agent ne sont pas encore bien définies. Aucune de ces méthodes n'a spécifié de façon explicite les modes de traitement, l'environnement de développement et les types d'interfaces qu'elles peuvent permettre.

En conclusion, la dimension technologie n'a pas été véritablement prise en compte par ces méthodes ou du moins est encore à un état embryonnaire.

7.2 Discussion générale

Les techniques des méthodologies évaluées ici constituent soit une extension des techniques des méthodologies orientées-objet, soit une extension des techniques des méthodologies à base de connaissance. En effet, les méthodes GAIA, MaSE, HLIM, MMTS et AOMEN utilisent comme toile de fond des techniques orientées-objet. L'avantage de ces méthodes est que l'expérience et le succès liés à l'utilisation des méthodes orientées-objet peuvent faciliter l'intégration de la technologie agent. De plus, elles offrent une base d'analyse intéressante. Les méthodes CoMoMAS et MAS-CommonKADS constituent une extension de la méthode à base de connaissance CommonKADS. MAS-CommonKADS est une combinaison des méthodes CommonKADS, OOSE, OMT et des langages SDL et MSC96 pour la spécification des protocoles des interactions des agents.

La méthode Cassiopée a été développée pour un contexte particulier, celui des robots footballeurs.

Les méthodes constituant une extension des méthodologies à base de connaissance et la méthode MMTS fournissent des modèles qui prennent mieux en compte l'état interne des agents. Cependant, ces modèles sont complexes (le nombre d'interrelations des modèles étant 10) ainsi que ceux de la méthode MASB.

Aucune des méthodes n'a fourni des modèles qui recouvrent totalement l'univers du discours. Le support logiciel (UCMs) utilisé par HLIM pour la représentation est très intéressant pour la compréhension, l'interprétation et la communication entre les différents acteurs du développement des SMA. Le formalisme utilisé par MaSE rend générique ses composantes et facile la vérification pendant les phases de spécification et de conception.

La dimension coopération et la dimension organisation restent encore, à notre avis, à débroussailler dans ces différentes méthodologies.

Enfin, la dimension technologie n'a pas été véritablement prise en compte par ces méthodes.

Suite à cette étude comparative des méthodologies ci-dessus citées, qui sont représentatives des méthodologies SMA existantes, il nous semble qu'une méthode idéale pour la conception des SMA devrait recouvrir tous les critères des six dimensions de notre cadre CaMuCCoSMA. Nous avons donné une description du cadre CaMuCCoSMA dans la syntaxe CML (voir annexe).

Chapitre 8

Quelques suggestions en vue de la conception d'une méthodologie SMA complète

8.1 Les suggestions

Après l'analyse comparative des 9 méthodologies SMA, nous proposons quelques suggestions préliminaires pour la conception d'une méthodologie SMA relativement complète. Ces suggestions comportent 7 points qui vont permettre, à notre avis, de mieux orienter la conception d'une méthodologie SMA.

1- *Une bonne méthodologie de développement de SMA devrait couvrir au mieux toutes les six dimensions du cadre CaMuCCoSMA.*

Nous suggérons ce point car nous pensons que le cadre CaMuCCoSMA essaye vraiment d'unifier toutes les étapes de la conception d'un SMA. Le fait de prendre en compte toutes les dimensions de CaMuCCoSMA apporte un plus que les autres méthodes ne possèdent pas.

2- *Nous suggérons pour la Dimension Méthodologie, la méthode MAS-CommonKADS* [CAR98].

En effet, MAS-CommonKADS couvre mieux que les autres méthodes, les étapes de la conception d'un SMA. Les résultats présentés dans le tableau 7.1 au chapitre 7 de ce document en sont une bonne justification. Elle est conçue avec la réutilisation et pour la réutilisation, en ce sens qu'elle utilise une grande variété de techniques préexistantes provenant soit de la technologie objet, soit des méthodes à base de connaissance et que ses modèles sont génériques. Le modèle spiral utilisé par MAS-CommonKADS pourrait être remplacé par le *modèle nabla* [KOS99] afin de mieux caractériser les interfaces hommes-machines et de mieux prendre en compte le point de vue de l'utilisateur.

3- *Pour la Dimension Représentation*

Nous suggérons de voir comment transposer les concepts et formalismes utilisés par MAS-CommonKADS dans la notation des UCMs utilisée par la méthode HLIM [EAL99]. Cette notation va permettre de diminuer la complexité des modèles de MAS-CommonKADS (se

référer à la section 8.2). Afin de permettre la vérification des SMA, on pourrait voir comment intégrer les concepts des langages de spécifications AgML et AgDL utilisés dans la méthode MaSE [SCO99]. En effet, on peut utiliser les diagrammes de communication pour décrire les conversations entre agents. Ces diagrammes utilisent des machines d'états finis qui capturent la dynamique au niveau de l'échange des messages entre les agents. Ces machines d'états finis débouchent sur une spécification algébrique des conversations, ce qui permet de construire des preuves formelles pour valider les interactions entre les agents et donc le système multi-agents. Il faut noter que les conversations entre agents constituent le support physique de la coopération et de la coordination dans les SMA.

4- *Pour la Dimension Agent*

Nous proposons d'utiliser les modèles d'expertise et d'agent de la méthode MAS-CommonKADS et le modèle d'agent de la méthode MASB. En effet, le modèle d'expertise et le modèle d'agent de MAS-CommonKADS prennent mieux en compte les caractéristiques intrinsèques des agents. Le modèle d'agent de MASB prend mieux en compte les trois positions de Dennett [DEN87] au niveau des attributions des agents du cadre CaMuCCoSMA. On peut utiliser les informations de ces modèles pour bien satisfaire les concepts d'agent cités dans la dimension agent du cadre CaMuCCoSMA (voir la section 6.2.3 du chapitre 6).

5- *Pour la Dimension Organisation*

Nous proposons d'utiliser, entre autres, les concepts organisationnels définis dans [ZAM00]. Ces concepts nous paraissent plus formels et permettent la conception des systèmes ouverts. Parmi ces concepts, on peut citer :

- Les règles de l'organisation : c'est l'ensemble des règles génériques qui régissent l'organisation.
- Les structures organisationnelles : ces structures définissent les classes spécifiques de régimes d'organisation et de contrôle auxquels les agents doivent se conformer dans tout le SMA pour travailler de façon efficace selon les objectifs. [ZAM00] a essayé de donner des définitions formelles à ces concepts. Par exemple, soient $\Re$ désigne l'ensemble des rôles des agents d'un système et $\Re_c$ la relation de contrôle entre ces

rôles. Un rôle r contrôle un rôle r' si r' exécute doit exécuter n'importe quel service demandé par r. On a les propriétés suivantes :

- $\mathfrak{R}_c \subseteq \mathfrak{R} \times \mathfrak{R}$
- $\mathfrak{R}_c$ est réflexive c'est–à-dire que $\forall$ r$\in \mathfrak{R}$, (r, r)$\in$ $\mathfrak{R}_c$
- $\mathfrak{R}_c$ n'est pas symétrique c'est-à-dire que si (r, r')$\in$ $\mathfrak{R}_c$ alors (r', r)$\notin$ $\mathfrak{R}_c$
- $\mathfrak{R}_c$ est transitive c'est-à-dire que si (r, r')$\in$ $\mathfrak{R}_c$ et si (r', r")$\in$ $\mathfrak{R}_c$ alors (r, r")$\in \mathfrak{R}_c$

6- *Pour la Dimension Coopération*

Nous suggérons d'exploiter les principes de réorganisation dynamique des groupes utilisés dans la méthode Cassiopée. Ces principes rendent l'interaction entre agents dynamique et permettent de résoudre des états non coopératifs. En effet, la mise en œuvre d'un comportement de résolution individuel affecte et est affectée par les comportements mis en œuvre par d'autres agents. Il y a donc des dépendances fonctionnelles inhérentes à l'accomplissement collectif de la tâche considérée. L'ensemble de ces dépendances détermine le couplage du problème d'organisation sous-jacent à l'accomplissement collectif de la tâche considérée. Les auteurs de Cassiopée distingue deux types de couplage :

1. le couplage statique (lorsqu'il n'y a pas de concurrence entre les comportements individuels). Dans ce cas, l'organisation entre les agents ne change pas;
2. le couplage dynamique (quand il y a une forme de concurrence à gérer; par exemple, il existe des comportements individuels équivalents pour une situation donnée ou un même comportement peut être mis en œuvre par différents agents). Dans ce cas, l'organisation dépend du contexte dans lequel se trouve les agents. C'est à ce deuxième cas que s'intéresse la méthode Cassiopée où les robots footballeurs doivent s'organiser dynamiquement pour résoudre collectivement leur tâche. Le concepteur ne peut alors définir à priori l'organisation entre agents comme dans le cas d'un couplage statique. Il ne peut donc s'intéresser aux structures organisationnelles entre les agents. Il définit ces structures en utilisant les concepts de "graphe de couplage" et d'"influence". Un graphe de couplage contient les dépendances entre rôles qui sont jugés pertinentes pour l'accomplissement collectif de la tâche. Les chemins et les circuits élémentaires de ce graphe de couplage définissent les possibilités de

regroupement des différents rôles du domaine et fournissent ainsi une représentation globale de la structure des organisations auxquelles les agents, en jouant ces rôles, peuvent appartenir [ANE96]. Pour que les agents soient capables de déterminer leur rôle en fonction de ceux joués par les autres agents, Cassiopée utilise la notion d'influence. La notion d'influence est un moyen permettant aux agents d'identifier et de traiter les dépendancs. Il existe une relation d'influence entre un agent A et un agent B, s'il existe une dépendance entre le rôle joué par A et celui joué par B [ANE96].

Pour supporter et renforcer la conception des SMA ouverts, on peut intégrer le modèle d'interaction dynamique (MID) de [RIB00]. Ce modèle permet l'effacement et l'intégration dynamique des agents au sein d'un SMA. Aussi, le concept de message actif utilisé dans le MID permet de décharger les agents du système des lourds protocoles d'interaction entre agents, ce qui permet aux agents de se concentrer uniquement sur leur tâche spécifique et d'être plus efficaces.

7- *La Dimension Technologique est un axe qui peut prendre assez de temps.*

Nous pensons que les logiciels visés devraient vérifier les critères définis pour cette dimension dans le cadre CaMuCCoSMA.

8.2 Étude de cas : Agence de voyage

Une justification complète de ces suggestions reviendrait à développer une méthodologie SMA. La conception d'une méthodologie n'est pas une chose si facile et peut prendre plus de temps que celui disponible pour réaliser ce mémoire dans le cadre de notre maîtrise. À défaut de donner une justification complète, une démarche pour justifier la faisabilité de ce que nous proposons comme suggestions est la suivante :

- définir le cadre méthodologique que doit avoir la méthodologie qu'on veut concevoir (nous suggérons de prendre le cadre méthodologique de MAS-CommonKADS). On doit à cette étape identifier les modèles nécessaires pour concevoir un SMA ou mieux, ce cadre doit suivre les critères définis dans la dimension méthodologique de CaMuCCoSMA. Le modèle de développement devant être le modèle nabla [KOL97];

- la modélisation doit être faite en utilisant la notation des UCMs;
- définir le modèle d'agent du système en utilisant les modèles d'expertise et modèle d'agent de MAS-CommonKADS, et le modèle d'agent de MASB. Ce modèle d'agent doit posséder les caractéristiques de la dimension agent du cadre CaMuCCoSMA. Chacune de ces caractéristiques sera la meilleure tirée des trois modèles de MAS-CommonKADS et de MASB, ci-dessus cités;
- définir les règles et les structures organisationnelles que doit avoir le SMA (auquel la méthodologie peut s'appliquer) et les caractéristiques de l'environnement du SMA en utilisant le formalisme proposé dans [ZAM00]. Ces règles, structures et caractéristiques doivent être définies de façon à ce qu'on puisse les instancier à chaque type de SMA qu'on peut concevoir avec la méthodologie;
- prendre le modèle d'interaction dynamique (MID) de [RIB00] comme modèle d'interaction de la méthodologie qu'on veut concevoir. Si certains principes de réorganisation dynamique des groupes de la méthode Cassiopée sont manquants dans le MID, on peut les y ajouter.

Cependant, les résultats issus de l'étude comparative des méthodologies SMA ici étudiées (chapitre 7) constituent une sorte de justification de ces suggestions puisqu'elles sont produites à partir de cette étude. Pour compléter un tant soit peu cette justification nous nous proposons de montrer la faisabilité du troisième point concernant la dimension représentation par une étude de cas de la documentation supportant la méthodologie MAS-CommonKADS. Nous laissons la partie vérification de ce troisième point car elle nécessite un long processus. Il est à signaler que même les auteurs de MaSE n'ont pas encore complètement défini les langages AgML et AgDL [SCO99]. Pour faire une telle justification, nous allons considérer le formalisme selon MAS-CommonKADS; ensuite le même exemple sera représenté en utilisant les UCMs et une conclusion suivra.

8.2.1 Définition du problème

Il s'agit de concevoir un système qui sera consulté par un usager pour la réservation d'un vol. Le système doit être en mesure de fournir à l'usager tous les vols disponibles avec leur plus faible probabilité d'être retardés. Le système est générique (instanciable) et doit pouvoir être

utilisé par n'importe quelle compagnie. Les informations sur les vols seront fournies à partir des lignes aériennes.

8.2.2 Formalisation selon MAS-CommonKADS

La phase de conceptualisation permet d'identifier un rôle de l'utilisateur, le Voyageur : une personne qui désire voyager. Le Système doit lui fournir les informations suivantes :

- la date de départ (dd);

- la date d'arrivée (da);

- la destination (dest).

Le diagramme des cas d'utilisation en Message Sequence Chart (MSC)) [EKK96] du Voyageur est donné par la figure 8.1.

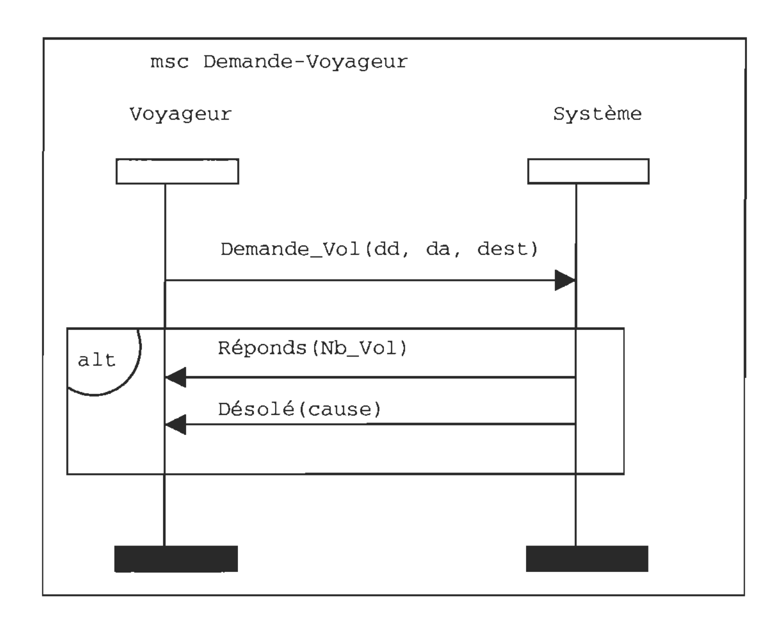


Figure 8.1 : Diagramme des cas d'utilisation de la requête du Voyageur

Le Voyageur interroge le Système sur les vols disponibles avec les arguments suivants : dd, da et dest.

Le Système peut retourner comme réponse : vols disponibles avec leurs numéros de vol ou pas de vol disponible en indiquant la raison. Dans le second cas, le Voyageur peut changer les données entrées. Le commentaire "alt" signifie que les deux réponses possibles se donnent de façon alternative par le Système.

La phase d'analyse permet d'identifier les types d'agents possibles pour le Système. Les types d'agents identifiés sont :

- Agent Interface (Secrétaire) : un agent humain qui joue le rôle d'interface entre le Système et le Voyageur;
- Prédictor : un agent dont le rôle est de prédire les vols sans retard;
- AgentLA : un agent commis dont le rôle est de fournir les informations sur les lignes aériennes. Cet agent peut être constitué d'un groupe d'agents.

La figure 8.2 donne le diagramme interne des cas d'utilisation en MSC (Message Sequence Charts) [EKK96] du Système. Ce diagramme décrit le prototype des scénarios entre les agents du Système.

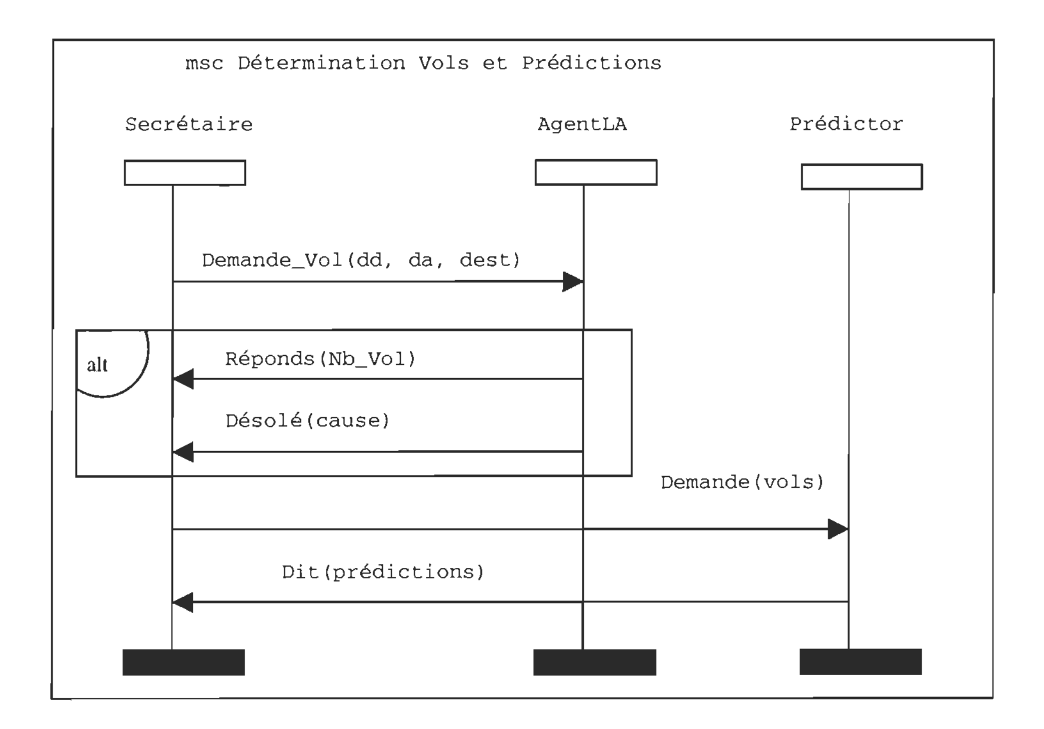


Figure 8.2 : MSC Diagramme interne des cas d'utilisation

Le diagramme de flux d'événements (Figure 8.3) représente les événements (messages échangés) entre les agents. Ce diagramme donne une vue sur les rapports inter-agents, du point de vue services.

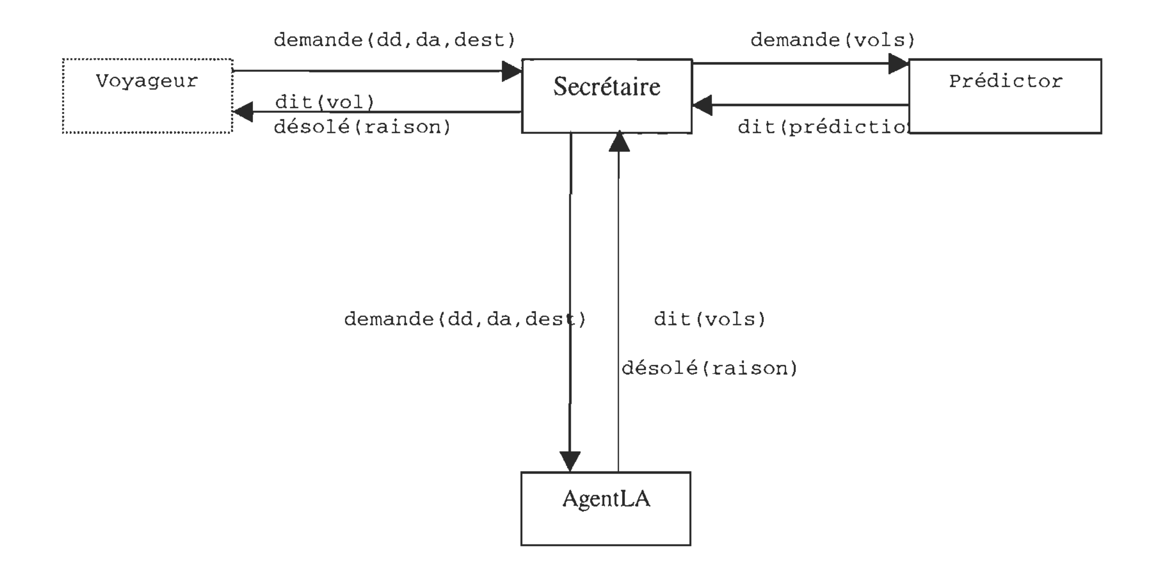


Figure 8.3 : Diagramme du flux d'événements

Ces deux derniers diagrammes sont des diagrammes du modèle de coordination de MAS-CommonKADS.

8.2.3 Formalisation avec les UCMs

Le Voyageur demande (dem) au Secrétaire les informations sur les vols disponibles. Le Secrétaire à son tour s'adresse simultanément au Prédicateur et à l'AgentLA. Il demande à l'AgentLA de lui fournir (demv) des informations sur les lignes aériennes et au Prédictor de prédire (demp) les vols sans retard.

Le Prédictor prédit (Pred) les vols sans retard et les communique (com1) au Secrétaire. L'AgentLA consulte (cons) sa base de données (Bd) des lignes aériennes et communique (com2) les résultats au Secrétaire en fonction du fait qu'il y ait des vols disponibles ([oui]) ou pas ([non]). Après avoir reçu les réponses provenant du Prédicateur et de l'AgentLA, le Secrétaire communique (com3) les résultats intégrés au Voyageur. Si le Voyageur est satisfait, le processus s'arrête; dans le cas contraire il peut réinitier le processus en changeant les données entrées.

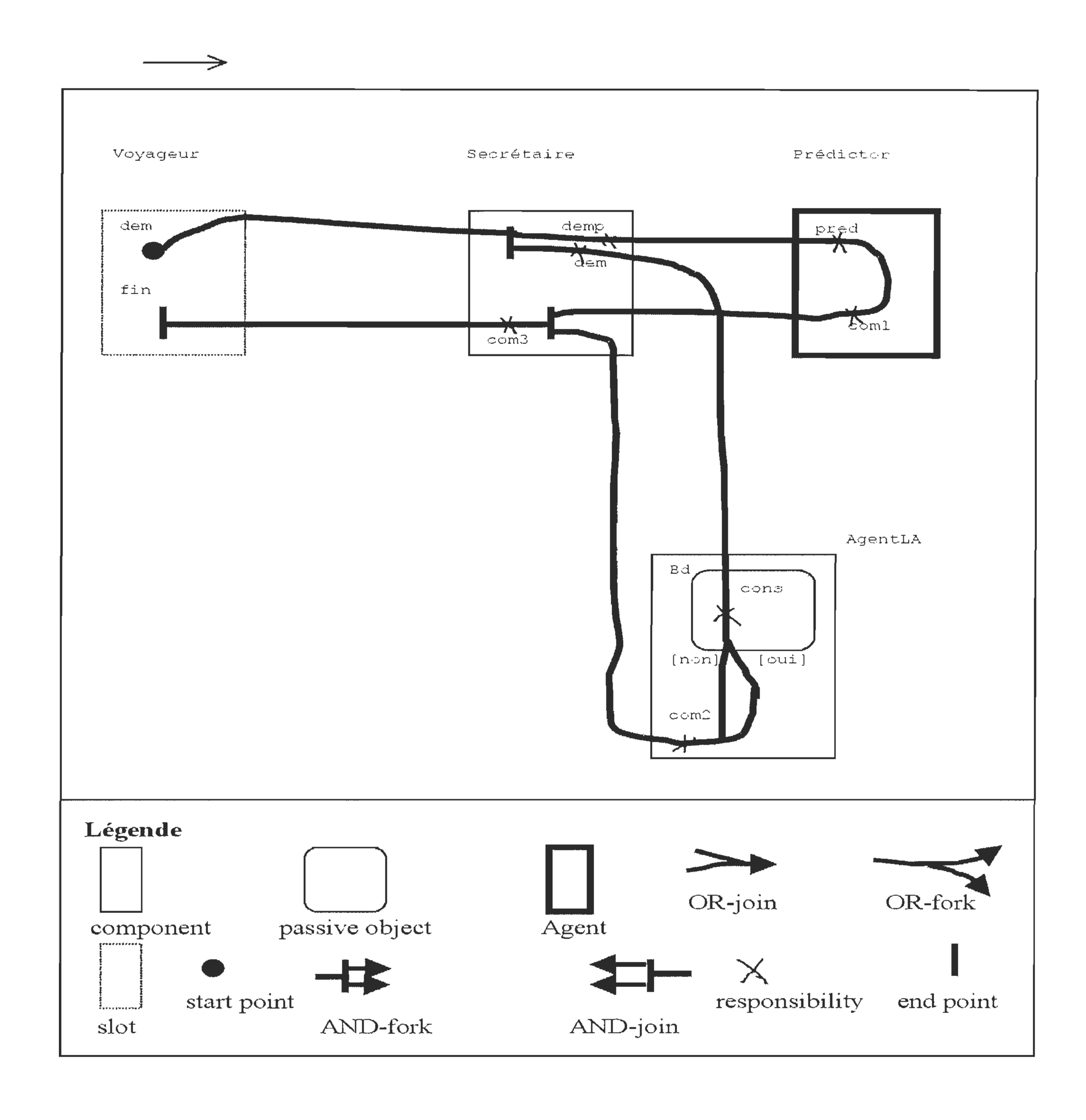


Figure 8.4 : Diagrammes en UCMs

Légende

- Start point représente des pré conditions ou des déclencheurs;
- End point représente des post conditions ou des résultats;
- Component représente une composante ou un module du système; lorsqu'une responsabilité (une action, une tâche ou une fonction) se trouve à l'intérieur d'une composante, on dit que cette composante est en charge de l'exécution de cette responsabilité;
- Passive object représente un objet;
- Agent a le même sens que celui donné dans ce document;
- Slot est un conteneur de composantes dynamiques (des composantes qui peuvent être créées, déplacées et supprimées pendant l'exécution);
- OR-join fusionne deux chemins arrivants;
- OR-fork divise un chemin en deux;
- AND-join synchronise deux ou plusieurs chemins;
- AND-fork divise un chemin en deux ou plusieurs segments concurrents;

8.2.4 Discussion sur les deux représentations

Nous pouvons observer que la figure 8.4 combine ensemble les figures 2 et 3 dans une forme compacte. De plus, elle spécifie les différentes responsabilités nécessaires au fonctionnement du système (c'est-à-dire les comportements de chacune des composantes du système); ce que les diagrammes de MAS-CommonKADS ne montrent pas. Ces diagrammes spécifient seulement les messages échangés entre les composantes du système.

Cet exemple simple montre que l'utilisation des UCMs peut diminuer la complexité des modèles de la méthode MAS-CommonKADS, en réduisant son nombre de modèles et en condensant beaucoup d'informations dans une forme plus compacte.

8.3 Conclusion

Cette section a présenté quelques suggestions préliminaires pour la conception des méthodologies SMA. Ce que nous proposons n'est pas une méthodologie mais plutôt une piste d'unification des méthodologies ici étudiées. Nous avons proposé ces suggestions sur la base des résultats issus de l'étude comparative de ces méthodologies faite dans le chapitre 7. Elles serviront éventuellement de ligne directrice pour la conception d'une méthodologie relativement

complète de développement de SMA. Nous n'avons pas la prétention de blâmer les méthodologies SMA existantes. Notre travail se situe dans un contexte plutôt d'amélioration pour arriver un jour à quelque chose de mieux.

Chapitre 9

Conclusion

Nous espérons, avec notre mémoire, contribuer à rendre plus systématique la conception des applications SMA et participer à la convergence et à la standardisation des concepts et des plateformes SMA. Nous pensons que la convergence des méthodes SMA peut indirectement avoir des effets positifs sur la standardisation des plateformes SMA.

Le cadre CaMuCCoSMA (Cadre Multidimensionnel de Critères pour la Comparaison des Méthodologies SMA) que nous avons proposé a été élaboré par l'importation et l'adaptation des concepts de la technologie objet et du Génie Logiciel au contexte des SMA. Nous avons aussi exploité des travaux de certains auteurs (se référer à la section 6.1 du chapitre 6). Le cadre CaMuCCoSMA contient six dimensions. Chaque dimension est définie avec des critères. Chaque critère a un certain nombre de valeurs. Nous avons donné une sémantique (au regard du contexte des SMA) aux critères que nous avons importés des travaux d'autres auteurs, mais également, nous avons ajouté des critères pour ce cadre. Ce cadre nous a permis de faire une étude comparative des principales méthodologies SMA existantes. À travers cette étude, nous avons dégagé les avantages qu'apporte chacune de ces méthodologies à la conception des SMA, les points communs à ces méthodologies et surtout leurs insuffisances. Avec les résultats de cette analyse, nous avons proposé quelques suggestions préliminaires qui serviront (dans le futur) de lignes directrices pour la conception d'une méthodologie relativement complète de développement de SMA. Ce que nous proposons n'est pas une méthodologie mais plutôt une piste d'unification (dans le même esprit que UML) des méthodologies ici étudiées. Ces suggestions contiennent 7 points prenant en compte les dimensions du cadre CaMuCCoSMA.

Dans l'annexe, nous donnons une description du cadre CaMuCCoSMA dans la syntaxe du langage CML (Conceptual Modeling Language) [SCH94a]. Cette syntaxe rend simple l'exploitation du cadre CaMuCCoSMA et la communication entre les acteurs du développement de SMA.

Il est à signaler que ce travail de recherche a fait l'objet d'une publication [SAB01a] et d'une communication au 69e congrès de l'ACFAS [SAB01b].

Annexe

Description du Cadre CaMuCCoSMA à l'aide du langage CML

Nous utilisons la syntaxe du langage CML (Conceptual Modeling Language) [SCH94a] pour décrire le cadre CaMuCCoSMA. Il reflète exactement les dimensions du cadre CaMuCCoSMA.

```
Méthodologie ::= MÉTHODOLOGIE   Nom_Méthodologie ;
                  Dimension Méthodologie
                  Dimension Représentation
                  Dimension Agent
                  Dimension Organisation
                  Dimension Coopération
                  Dimension Technologie
                Fin MÉTHODOLOGIE  [Nom_Méthodologie ] ;


Dimension Méthodologie ::= DIMENSION METHODOLOGIE ;
          Étapes du processus : analyse | modélisation |
          spécification | conception | validation |
          vérification| évaluation ergonomique;
          Modèles de développement : cascade| V | spirale |
          incrémental | nabla;
          Approches de développement : descendante |
          ascendante | évolutive;
          Degré d'implication de l'utilisateur : faible |
          moyen | fort;
```

Moment d'implication de l'utilisateur : début | milieu | fin;

Réutilisabilité;

Disponibilité de supports logiciels :

Analyse;

Modélisation;

Spécification;

Conception;

Validation;

Vérification;

Évaluation ergonomique.

Fin DIMENSION METHODOLOGIE ;

Dimension Représentation ::= DIMENSION REPRESENTATION ;

Découpage du système : (moyens d'assemblages)

Formalisme;

Séquencement des modèles;

Qualité des modèles;

Nombre de modèles;

Complétude des modèles;

Complexité des modèles : nombre d'interrelations entre les modèles.

Fin DIMENSION REPRESENTATION ;

Dimension Agent ::= DIMENSION AGENT;

Nature des agents : homogènes | hétérogènes.

Types d'agents : intelligents | interface | mobiles | d'information |autonomes;

Attributs des agents : adaptabilité | autonomie | comportement coopératif | capacité déductive | habileté de communication | mobilité |

personnalité |réactivité | continuité temporelle| comportement délibératif;
Attributions des agents : Position physique| stratégie de conception | stratégie intentionnelle.
Fin DIMENSION AGENT;

Dimension Organisation ::= DIMENSION ORGANISATION ;
Image d'organisation : système hiérarchique | système distribué | système holonique | système ouvert;
Nature de l'environnement : structuré | stable | déterministe |explicite | observable;
Type de l'environnement : actif | passif;
Caractéristiques des données : quantitative | qualitative.
Fin DIMENSION ORGANISATION ;

Dimension Coopération ::= DIMENSION COOPÉRATION ;
Type de communication : agents hétérogènes | agents humains;
Mode de communication : direct | indirect | synchrone |asynchrone;
Langage de communication : signaux | actes de discours | autre;
Modèle de coopération : négociation | délégation de tâches | planification multi-agents.
Type de contrôle : centralisé | hiérarchique | distribué;
Interaction : Statique | Dynamique;
Moteur d'interaction : distribué | centralisé;
Protocoles d'interaction : explicite | implicite;

Mécanisme d'interaction résout états non coopératifs : oui| non.

Fin DIMENSION COOPÉRATION;

Dimension Technologie ::= DIMENSION TECHNOLOGIE;

Mode de traitement : batch | interactif | client-serveur |synchrone | asynchrone | distribué;

Type d'interface homme-machine : classique | adaptable| adaptative | assistance;

Programmation : structurée | orientée-objet | orientée agent;

Type d'application visée : Simulation | Résolution de problème | Intégration.

Fin DIMENSION TECHNOLOGIE ;

Références

[ADA99a] Adam E., Kolki C., Étude comparative de méthodes de génie logiciel en vue du développement de processus administratifs complexes, 1999.

[ADA99b] Adam E., Mandiau R., Kolski C., Approche holonique de modélisation d'une organisation orientée workflow : SOHTCO 1999.

[ADA99c] http ://perso.wanadoo.fr/famille.adam/eadam/publis.htm

[AFI98] AFIA/PRC-13 Systèmes Multi-agents, Architectures de Systèmes d'Agents

[ALL94] Alliot Jean-Marc, Schiex Thomas, Intelligence Artificielle et Informatique Théorique ; CEPAD ; Mars 1994.

[AMY99a] Amyot, D., Use Case Maps Quick Tutorial Version 1.0 02-Sep-99.

[AMY99b] Amyot, D. and Miga, A., Use Case Maps Linear Form in XML, version.

[ARN98] Arnold, K. and Gosling, J., The Java Programming Language. Addison-Wesley Publishing Co., second edition. 1998.

[BAE98] Baeijs Christof, Fonctionnalité émergente dans une société d'agents autonomes. Étude des aspects organisationnels dans les SMA réactifs. Thèse de doctorat 1998.

[BAR95] Barbuceanu, M. and Fox M. S., «COOL; A Language for Describing Coordination in Multi-Agent Systems», First International Conference on Multi-Agent Systems, pp. 17-24, 1995.

[BER93] Berlage Thomas, Object-oriented application frameworks for graphical user interfaces : the GINA perspective. Gesellschaft für Mathematik und Datenverarbeitung, München, Wien, Oldenbourg, 1993.

[BRA85a] Bravoco R. R., and Yadav S. B., "Requirement Definition Architecture – An Overview", Computers in Industry, Vol.6, pp. 237- 251, 1985.

[BRA85b] Bravoco R. R., and Yadav S. B., "A methodologie to Model the Functional Structure of Organization", Computers in Industry, Vol. 6 pp. 345-361, 1985.

[BRA97] Bradshaw J. M., An Introduction to Software Agents, In : Software Agents, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 3-46.

[BRA97] Brazier F.M.T., Dunin-Keplicz B. M., Jennings N. R., and Treur J., DESIRE : Modeling multi-agent systems in a compositional formal framework. Int Journal of Cooperative Information Systems, 1(6) : 67-94, January 1997.

[BUR98a] Buhr R.J.A., Elammari M., Gray T., and Mankovski S., A High Level Visual Notation for Understanding and Designing Collaborative, Adaptive Behaviour in Multiagent Systems. 1998. Artificial Intelligence. Wiley-Interscience Publications. pp. 3-55. 1996.

[CAL99] Calderoni, Thèse de Doctorat de l'Université de la Réunion, Novembre 1999.

[CAR98] Carlos A.I., Mercedes Garjo, José Gonzàlez C., and Velasco Juan R., Analysis and Design of Multiagent Systems Using MAS-CommonKADS. In AAA'97 Workshop on Agent Theories, Architectures and Languages, Providence, RI, July 1997. ATAL. An extended version of this paper has been published in INTELLIGENT AGENTS IV : Agent Theories, Architectures, and Languages, Springer Verlag,1998.

[CAR99] Carlos A.I., Mercedes G., Jose C. G., A Survey of Agent-Oriented Methodologies. Computer Science, V. 1555, p.317, 14p. 1999.

[CHA99] Chaïb-Draa Brahim, Agents et Systèmes Multiagents (IFT 64881A). Notes de cours. Département d'informatique, Faculté des sciences et de génie, Université Laval, Québec. Novembre 1999.

[CNR01] http ://www.cnrs.fr/

[COH94] Cohen P.R., Cheyer A., Wang M., and Baeg S.C., An open agent architecture. In : Proceedings of the AAAI Spring Symposium. 1994.

[COL94] Colman D., Arnold P., Bodoff S., Dollin C., Gilchrist H., Hayes F., and Jeremaes P., Object-Oriented Development : The FUSION Method. Prentice-Hall International : Hemel Hempstead England, 1994.

[COL95] Collinot Anne, Carle P., and Zeghal K., Cassiopée : a Method for Design Computational Organizations. In Proceedings of the First International Workshop on Decentralized Intelligent Multi-Agent Systems, 124-131. Krakow, Poland, 1995.

[COL96] Collinot Anne, Drogoul Alexis, and Benhamou Philippe. Agent oriented design of a soccer robot team. In Proceedings of the Second International Conference on Multi-Agent systems(ICMAS-96),pages 41-47, Kyoto, Japan, December 1996.

[CON98] W3 Consortium : Extensible Markup Language (XML) 1.0. W3C Recommendation, February 1998. http ://www.w3.org/TR/REC-xml

[COR01] http ://www.cirad.fr/presentation/programmes/espace/cormas/eng/index.shtml

[COT99] Côté Marc, Une architecture multiagent et son application aux services financiers. Mémoire présenté à la faculté des études supérieures de l'Université Laval. Faculté des sciences et de génie. Département d'informatique. Avril 1999.

[DEC87] Decker K. S., Distributed Problem-Solving Techniques : A survey. IEEE Transactions on Systems, Man and Cybernetics 17(5) : 730-740, 1987.

[DEC96] Decker, K., Williamson, M. and Sycara, K., Matchmaking and Brokering. In : Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), December, 1996.

[DEN87] Dennett, D.C., The Intentional Stance. MIT Press, Cambrige, Mass.1987.

[DIM01] http ://www-poleia.lip6.fr/~guessoum/dima.html

[DUR87] Durfee E. H. and Lesser V. R., Using partial global plans to coordinate distributed problem solvers. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87), Milan, Italy, 1987.

[DUR89] Durfee E. H. and Lesser V. R., Negotiating task decomposition and allocation using partial global planning. In L. Gasser and M. Huhns, editors, Distributed Artificial Intelligence Volume II, pages 229-244. Pitman Publishing : London and Morgan Kaufmann : San Mateo, CA, 1989.

[EAL99] Ealmmari M., Lalonde W., An Agent-Oriented Methodology : High-Level and Intermediate Models. Proceedings of AOIS-1999. Heidelberg (Germany), June 1999.

[EGO01] Egon Verharen, Frank Dignum, and Sander Bos, Implementation of a cooperative agent architecture based on the language-action perspective (In this volume).

[EGO97] Egon M. Verharen, A language-Action Perspective on the Design of Cooperative Information Agents. Ph.D thesis, Katholieke Universieit Brabant, the Netherlands, March1997.

[EKK96] Ekkart Rudolph, Grabowski Jens, and Graubmann Peter, Tutorial on message sequence chart (MSC). In Proceedings of FORTE/PSTV'96 Conference, October 1996.

[FER95] Ferber J., Les systèmes Multi-Agents : vers une intelligence collective. InterEdition, Paris, France, 1995.

[FIN93] Finin T. et al, Specification of the KQML Agent-Communication Language. Draft. June 1993.

[FIP93] FIPS Pub183, Integration definition function modeling (IDEF0). Software Standard Modeling Techniques. FIPS Pub183, Computer Systems Laboratory National Institute of Standards and Technology, Gaithersburg, Md. 20899,1993.

[FOX81] Fox M.S., An organizational view of distributed systems. IEEE Trans. Syst.Man. Univ. Cybern., vol. SMC-11; 1981, pp. 70-80.

[GEN97] Genesereth M., An Agent-based Framework for Interoperability. In : Software Agents, J.M. Bradshaw (Ed.), Menlo Park, Calif., AAAI Press, 1997, pages 317-345.

[GEO90] Georgeff M. P., and Ingrand F., «Research on Procedural Reasoning Systems», SRI International, 1990.

[GEO95] Geomed, Geomed-F (IE feasibility project no 174) : Final report. Technical report, GMD, Sankt Augustin, Germany, 1995.

[GLI98] Glize Pierre, Gleizes Marie-Pierre et Camps Valérie, Une théorie de l'apprentissage fondée sur l'auto-organisation par coopération. 1998.

[GOL94] Goldman C.V., Rosenschein J.S., «Emergent Coordination through the Use of Cooperative Stage-Changing Rule», Proceedings of the Twelve National Conference on Artificial Intelligence, Seattle, Washington. 1994.

[GRA01] http ://www.granddictionnaire.com/_fs_global_01.htm

[GRU92] Gruber T. R, «Ontolingua : A Mechanism to Support Portable Ontologies», Reference Manual, June, 1992.

[GUE01] http ://www-poleia.lip6.fr/~guessoum/asa/afia

[HAR87] Harel D., Statecharts : A visual formalism for complex systems. Sci. Computer Program, 8 : 231-247. 1987.

[HOG92] Hogg T., Huberman B.A., Better than the best : The power of cooperation, Lectures in complex systems, Addison Wesley.

[HUH87] Huhns M., Mukhopadhyay U., and Stephens L. M., DAI for document retrieval : The MINDS project. In M. Huhns, editor, Distributed Artificial Intelligence, pages 249-284. Pitman Publishing : London and Morgan Kaufmann : San Mateo, CA, 1987.

[ITU94] ITU-T.Z100 (1993), CCITT specification and description language (sdl). Technical report, ITU-T, June 1994.

[JAC92] Jacobson I., Christerson M., Johnson P., and Overgaard, Object-Oriented Software Engineering. A Use Case Approach. ACM Press, 1992.

[JEN96a] Jennings N. R., Faratin P., Norman T. J., O'Brien P., Wiegand M.E., Voudouris C., Alty J. L., Miah T., and Mamdani E. H. (1996), «ADEPT : Managing Business Processes Using Intelligent Agents», in : Proc. BCS Expert Systems 96 Conference (ISIP Track), Cambridge, UK 5-23.

[JEN96b] Jennings N. R., Corera J., Laresgoiti I., Mamdani E. H., Perriolat F., Skarek P. and Varga L. Z., «Using ARCHON to develop real-word DAI applications for

electricity», Transportation management and particle accelerator control, IEEE Expert, 1996, 11 (6).

[JEN98] Jennings N.R., Sycara K. and Wooldridge M., A Roadmap of Agent Research and Development.In : Autonomous Agents and Multi-Agent Systems Journal, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, 1998, Volume 1, Issue 1, pages 7-38.

[KAO01] http ://ksi.cpsc.ucalgary.ca/KAW/KAW96/bradshaw/KAW.html

[KEN96] Kendall Elisabeth A, Malkoun Malgaret T., and Chong Jiang, A methodology for developing agent based systems for enterprise integration. In D.Luckose and Zhang C., editors, Proceedings of the First Australian Workshop on DAI, Lecture Notes on Artificial Intelligence. Springer-Verlag :Heidelberg, Germany, 1996.

[KIN96] Kinny David, Georgeff Michael, and Rao Anand, A methodology and modeling technique for systems of BDI agents. In W. Van der Velde and J. Perram, editors, Agents Breaking Away : Proceedings of the Seventh European Workshop on Modelling Autonmous Agents in a Multi-Agent World MAAMAW'96, (LNAI Volume1038). Springer-Verlag : Heidelberg, Germany,1996.

[KOE 89] Koestler A., «The GHOST in the MACHINE», Editions ARKANA, Londres, 1989.

[KOL97] Kolski C., Interfaces Homme-Machine, application aux systèmes industriels complexes. Editions Hermes, Paris.1997.

[KOS93] Kosanke K., CIMOSA- A European Development for Enterprise Integration. IOS Press, 1993.

[KUW95] Kuwabara K., Ishida T., and Osato N., «AgentTalk : Methodologies and Multiagent Coordination Protocols with Inheritance,» submitted to Tools for Artificial Intelligence Conference 1995.

[LAB97] Labrou Y.; Finin T., A Proposal for a new KQML Specification. TR CS-97- 03. February,1997.

[LOG98] Logan Brian, Classifying Agent Systems. In AAAI Press. Software Tools for Developing Agents. Papers from the 1998 Workshop Technical Report WS-98-10.

[MAG01] http ://www.lifl.fr/SMAC/projects/magique/

[MAS01] http ://www.emse.fr/

[MER01] http ://www.limsi.fr/RS96FF/CHM/LC/

[MIG98] Miga A., Application of Use Case Maps to System Design with Tool Support. M.Eng. thesis, Dept. of Systems and Computer Engineering, Carleton University, Ottawa,Canada,1998. http ://www.UseCaseMaps.org/UseCaseMaps/ucmnav/

[MOC01] http ://www.ai.univ-paris8.fr/~abchiche/

[MON90] Montgomery T. A. and Durfee E. H., Using MICE to study intelligent dynamic coordination. In Proc. 2nd IEEE Conf. On Tools for AI, pages 438-444,1990.

[MOU96a] Moulin Bernard and Brassard Mario, A scenario-based design method and an environment for the development of multiagent systems. In D. Lukose and C. Zhang, editors, First Australian workshop on Distributed Artificial Intelligence, (LNAI volume 1087), pages 216-231. Springer-Verlag :Heidelberg, Germany,1996.

[MOU96b] Moulin Bernard, Chaib-draa B., An Overview of Distributed Artificial Intelligence. In : O'HARE, G. ; JENNINGS, N. (eds.) Foundations of Distributed.

[MUL97] Muller P., Instant UML. Wrox Press, Birminggham, UK, 1997.

[NII89] Nii H. P., Blackboard Systems In The Handbook of Artificial Intelligence, A. Barr, P.R. Cohen and E.A. Feingenbaum (Eds.), Addison-Wesley, New York, 1989, Volume IV, chapter XVI, pages 1-82.

[NOR96] Norbert Glaser, Contribution to Knowledge Modeling in a Multi-Agent Framework (the CoMoMAS Approach). Ph.D thesis, l'Université Henri Poincaré, Nancy I, France, November,1996.

[OSA01] http ://www.utc.fr/

[PAP90] Le Pape C., A combination of Centralized and Distributed Methods for Multi-Agent Planning and Scheduling. In Proceedings of the IEEE International Conference on Robotics and Automation, 488-493. Cincinnati, 1990.

[PAS93] Pascot D., Bernadas C., L'essence des méthodes : Étude comparative de six méthodes de conception de systèmes d'information informatisés. INFORSID'93 «Systèmes d'information, Systèmes à base de connaissances», Lille, 11-14 Mai 1993.

[PAT92] Patil R.; Fikes R.; Patel-Schneider P.; Mckay D.; Finin T.; Gruber T.; Neches R., The DARPA knowledge sharing effort : Progress report In : Principles of Knowledge Representation and reasoning : Proocedings of the Third International Conference, November 1992. (www.cs.umbc.edu/kqml/papers/kr92.ps)

[PES97] Pesty S., Brassac C., et Ferrent P., Ancrer les agents cognitifs dans l'environnement. In : Actes des 5e Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents. – La Colle sur Loop, Avril 1997.

[RAO95] Rao, A., private conversation, 1995.

[RIC00] Ricordel P. M. and Demazeau Y., "From Analysis to Deployment : a MultiAgent Platform Survey", 1st International Workshop on Enginnering Societis in the Agents World (ESAW), ECAI'2000, A. Ominici, R. Tolksdorf and F. Zambonelli (eds.), august, 2000, Berlin, Germany, LNAI n°1972, Artificial Intelligence, pp.93-105, Springer Verlag, 2000.

[ROB99] Roberto A. Flores-Mendez; Towards the standardization of Multi-Agent Systems Achitectures : An Overview. ACM Crossroads'special issue on Intelligent Agents, summer 1999. [RUM91] J. Rumbaugh, M. Blaha, W. Premerlani, and V. Lorensen F. Eddy. Object-Oriented Modeling and Design. Prentice-Hall, 1991.

[SAB01a] Sabas Arsène, Delisle Sylvain, Badri Mourad, "Vers une unification des méthodologies de développement des systèmes multiagents", *Actes des Journées francophones pour l'intelligence artificielle distribuée et les systèmes multi-agents*, Montréal (Québec, Canada), 12-14 novembre 2001, *à paraître*.

[SAB01b] Sabas Arsène, Badri Mourad, Delisle Sylvain, "Une étude comparative des méthodologies SMA", *69e congrès de l'ACFAS*, Sherbrooke, Mai 2001.

[SAU00] Sauvave Sylvain, étudiant au Ph.D., Université de Caen, France. Utilisation des patterns pour l'analyse et la conception de SMA. Conférence à l'Université Laval (Canada), 12-09-00.

[SCH94] Schreiber A. Th., Wielinga B. J., and Akkermans J. M., Van de Velde W., CommonKADS : A comprehensive Methodology for KBS Development. Deliverable DM1.2a KADS-II/M1/RR/UvA/70/1.1, University of Amsterdam, Netherlands Energy Research Foundation ECN and Free University of Brussels, 1994.

[SCH94a] Schreiber G., Wielinga B. J., Akkermans J.-M., Van de Velde W., and Anjewierden A., CML : The CommonKADS conceptual modeling language. In Proc. 8th European Knowledge Acquisition Workshop, pages 1-25, Hoegaarden, Belgium, 1994. LNAI Series, 867.

[SCH94b] Schreiber G., Wielinga B. J., de Hoog R., Akkermans H., and Van de Velde W., Commonkads : A comprehensive methodology for KBS development. IEEE Expert, 9(6) : 28-37, 1994.

[SCO99] Scott A. Deloach., Multiagent Systems Engineering : A Methodology And Language for Designing Agent Systems . 1999.

[SEK95] Sekaran M., Sen S., «To help or not to help», Seventeenth Annual Cognitive Sciences Conference, Pitsburg, Pennsylvannia. 1995.

[SIC95] Sichman J. and Demazeau Y., Exploiting Social Reasoning to Deal with Agency Level Inconsistency. In Proceedings of the First International Conference on Multi-Agent Systems, 352-359. San Francisco. 1995.

[SMI80] Smith R. G. and Davis R., The Contract Net Protocol : High-Level Communication and Control in a Distributed Problem-Solver. IEEE Transactions on Computers C29(12) : 1104-1113. 1980.

[SYC89] Sycara K., Multiagent Compromise via Negociation. In Gasser L. and Hunks M. eds, Distributed Artificial Intelligence II. San Mateo, California : Morgan Kaufmann. Chapter 6, 119-137. 1989.

[THE88] Theron Paul, Guide pratique du génie logiciel. Editions EYROLLES, 1988.

[VAN90] Vanderveken D., Meaning and Speech Acts. Volume I. Cambridge University Press. NewYork, 1990.

[WIE98] Wiederhold G., Mediators in the Architecture of Future Information Systems. In : Readings in Agents Edited by HUHNS, M. N. ; SINGH, M. P. . Morgan Kaufmann, San Francisco, 1998. (originalement : IEEE Computer 25(3) :38-49,1992.)

[WIR90] Wirfs-Brock R., Wilkerson B., and Wiener L., Designing, Object-Oriented Software. Prentice-Hall,1990.

[WOO00a] Jeennings N. R. and Wooldridge M. (2000), «Agent-Oriented Software Engineering» in Handbook of Technology (ed. J. Bradshaw) AAAI/MIT Press.

[WOO00b] Wooldridge M., Jennings N.R., and Kinny D. (2000), «The Gaia Methodology For Agent-Oriented Analysis and Design» Journal of Autonomous Agents and Multi-Agent Systems 3 (3) 285-312.

[WOO98] Woodridge M. and Jennings N. R., Pitfalls of Agent-oriented Development.In Proceedings of Second International Conference on Autonomous Agents (Agent98) è Minneapolis/St Paul, MN, May 1998.

[ZAM00] Zambonelli F., Jennings N. R., and Wooldridge M. (2000), «Organisational Abstrations for the Analysis and Design of Multi-Agent Systems» Proc. 1st Int. Workshop on Agent-Oriented Software Engineering, Limerick, Ireland, 127-141.

http ://www-poleia.lip6.fr/%7Eguessoum/asa.html, 1998.